University of Alberta

LEARNING FEED-FORWARD CONTROL FOR VISION-GUIDED ROBOTICS

by

Simon Léonard

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2008

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-46357-4
Our file Notre référence
ISBN: 978-0-494-46357-4

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

To Holly,
For every second I worked while away from you...

Abstract

Robots are expected to do increasingly challenging tasks in complex environments. Because of the complexity, robots must use their sensors to understand the nature of a task and to monitor its execution. Visual servoing proposes several methods for using visual feedback to control the execution of tasks. A drawback of visual servoing is that it controls the motion of a robot at the velocity level. Thus, visual servoing is neither compatible with robots that require position commands nor do they provide a natural interface for path planning in special Euclidean space.

This thesis proposes an image-based feed-forward control system that is based on a formulation that is similar to its feedback counterparts. Specifically, the system requires image-based errors that are derived from the image space. Instead of mapping the errors to velocities, however, the errors are mapped to variations of the robot's position through an interaction matrix called the visuomotor function. After deriving the functions of various tasks, this thesis proposes a method toapproximate the parameters of the visuomotor systems. The approximation is composed of two stages that are computed on-line. In the first stage, the parameters specific to a 3D point are estimated with an incremental least squares algorithm based on QR factorization. Then, these parameters are made available to neighboring 3D points by generalizing them in the stereo space.

Experiments were done in simulations and on real robots for the following tasks: gaze control of a robotic head, 3D translations, mobile robots and 6 degrees of freedom motion. Results were also compared with a feed-forward system based on 3D reconstructions from stereo images.

Acknowledgements

I would like to acknowledge my supervisor Martin Jägersand for supporting this research. I also want to thank the members, past and present of the Robotics Lab at the University of Alberta: Zhen Deng, Azad Shademan and Jonathan Klippenstein. I want to thank Elizabeth Croft from The University of British Columbia who has supported me during the last two years. I especially want to acknowledge two very good friends: Chris Parker and Pantelis Elinas. Your friendship made my grad student life enjoyable (and cynical). I will miss solving the world's problems around daily coffees. Finally I want to thank my family and Holly for their unconditional love and support.

Table of Contents

1 Introduction 1

1.1 Motivation and Purpose . . . 2

1.2 Objectives . . . 3

1.3 Contributions . . . 4

2 Background and Review 6

2.1 Visual Servoing Architectures . . . 6

2.1.1 Input: Visual System . . . 6

2.1.2 Output: Motor System . . . 6

2.2 Direct visual servoing . . . 7

2.3 Indirect Visual Servoing . . . 8

2.4 Image-Based Visual Servoing . . . 8

2.4.1 Control Laws . . . 10

2.4.2 Stability . . . 12

2.4.3 Image-Based Trajectories . . . 13

2.4.4 Camera Model . . . 13

2.4.5 Stereo IBVS . . . 13

2.4.6 Image Features . . . 14

2.5 Position-Based Visual Servoing . . . 14

2.5.1 Control Law . . . 15

2.5.2 Pose Estimation . . . 17

2.5.3 Hand-Eye Calibration . . . 20

2.5.4 Learning and Sensory Control . . . 21

3 Visuomotor Function 22

3.1 Visuomotor Function . . . 22

3.1.1 6DOF Motion . . . 24

3.1.2 Pan-Tilt Unit . . . 26

3.1.3 Translation . . . 29

3.1.4 Mobile Robot . . . 29

3.2 Solution to the Visuomotor Function 31
3.3 Estimation of the Visuomotor Parameters 33
3.3.1 Decoupling of the Visuomotor Parameters 34
3.4 Summary 34

4 Generalization and Function Approximation **35**
4.1 Upper Bound on $|S_{L_{\mathbf{P}} \cap R_{\mathbf{P}}}|$ 36
4.2 Instance-Based Learning 39
4.2.1 K-Nearest Neighbors 39
4.2.2 Locally Weighted Regression 39
4.2.3 Radial Basis Functions 39
4.2.4 Cerebellar Model Articulation Controller 40
4.3 Approximation of Θ_S 43
4.3.1 Vector of Basis Functions 43
4.3.2 Albus-Kaczmarz Learning Algorithm 43

5 Experiments **45**
5.1 Implementation Details 45
5.2 Pan-Tilt Unit 49
5.2.1 Simulations: Visuomotor Function 49
5.3 Translation 53
5.4 6 Degrees of Freedom 71
5.4.1 Simulation: Visuomotor Function 71
5.4.2 Simulations: 3D Reconstruction 75
5.4.3 Real 76
5.5 Mobile Robot 80

6 Conclusion **90**

Bibliography **93**

A Appendix **103**
A.1 Visuomotor Function 6DOF 103
A.2 CMAC: Approximation of the Visuomotor Parameters 105

List of Tables

1.1 Compared to other methods, the visuomotor function defines a control law that maps image-based errors to position commands. 3

2.1 Geometric reconstruction versus camera calibration. 19

5.1 Intrinsic camera parameters of the camera used in simulation. 50

5.2 Mean absolute error and standard deviation for 6 DOF with real data. 80

List of Figures

2.1 Direct visual servoing. 7
2.2 Direct visual servoing. 8
2.3 Image-based visual servoing 8
2.4 Chaumette conundrum. 10
2.5 Position-based visual servoing 15
2.6 Pose estimation. 15
2.7 Two views geometry. 18

3.1 Geometric model: Camera frames L and R are mounted on a stereo rig S. 23
3.2 Kinematics of a pan-tilt unit: A typical PTU involves a translation of its end-effector. 27
3.3 Modified Denavit-Hartenberg diagram of a PTU mounted with camera (frame C). . 27
3.4 Z-Y Euler angles. 28
3.5 Geometric model: Camera frames L and R relative to the world frame W. 30

4.1 Two 1×4 cameras with identical intrinsic and extrinsic parameters. The shaded areas delimit the 4 possible stereo codes. 36
4.2 Two 1×4 cameras with a translation between their coordinate frames. The regions $1, \ldots, 10$ represent the 10 possible stereo codes. 37
4.3 Two 1×4 cameras oriented in a way to use all 16 stereo codes. 38
4.4 Epipolar geometry. The point ${}^{S}P$ projects to ${}^{L}\mathbf{p}$ and ${}^{R}\mathbf{p}$. The epipolar line ${}^{L}\mathbf{e}$ constrains the coordinates of ${}^{L}\mathbf{p}$. 38
4.5 Example of a 2D CMAC. Each input $\mathbf{x} \in \mathbb{R}^2$ activates $T = 3$ basis functions. Each basis function b_t is associated to a weight w_t that is combined to approximate y. . . 41
4.6 Positioning of receptive fields. 41

5.1 Overlays for 2D images. Each color represent an overlay and each bold contour represents the selected tile on each overlay. The yellow contour represent the boundary from the union of all the selected tiles. 46
5.2 Combination for updates. 48
5.3 Configuration of the PTU used in the simulations. 49

5.4 Actual and estimate values of pan and tilt angles (only the last 80 iterations are illustrated for clarity reason) 51
5.5 Absolute error of pan and tilt angles. 52
5.6 Update map. The intensity of a pixel reflects the amount times one of its tile was selected. 53
5.7 Error statistics of 100 simulations. 54
5.8 Error statistics of 100 simulations. 55
5.9 Setup for the eye-in hand cameras used for the PTU and for the 6DOF experiments. 56
5.10 Real PTU: Pan and tilt angles. 57
5.11 Real PTU: Absolute errors. 58
5.12 Update map. The intensity of a pixel reflects the amount of updates. 59
5.13 Simulated translations along the X axis. 60
5.14 Simulated translations along the Y axis. 61
5.15 Simulated translations along the Z axis. 62
5.16 Mean and standard deviation of the absolute X error (100 simulations). 63
5.17 Mean and standard deviation of the absolute Y error (100 simulations). 63
5.18 Mean and standard deviation of the absolute Z error (100 simulation). 64
5.19 Evolution of the condition number of for one tile with simulated data. 64
5.20 Mean and standard deviation of the absolute X error with additional disturbances (100 simulations). 65
5.21 Mean and standard deviation of the absolute Y error with additional disturbances (100 simulations). 65
5.22 Mean and standard deviation of the absolute Z error with additional disturbances (100 simulation). 66
5.23 Error and standard deviation for X translations using 3D reconstruction (100 simulations). 66
5.24 Error and standard deviation for Y translations using 3D reconstruction (100 simulations). 67
5.25 Error and standard deviation for Z translations using 3D reconstruction (100 simulations). 67
5.26 Translation along the X axis. 68
5.27 Translation along the Y axis. 69
5.28 Translation along the Z axis. 70
5.29 Evolution of the condition number of for one tile with real data. 71
5.30 6 DOF simulations: Error and standard deviation for rotations around the X axis (100 simulations). 72

5.31 6 DOF simulations: Error and standard deviation for rotations around the Y axis (100 simulations). 72
5.32 6 DOF simulations: Error and standard deviation for rotations around the Z axis (100 simulations). 73
5.33 6 DOF simulations: Error and standard deviation for X translations (100 simulations). 73
5.34 6 DOF simulations: Error and standard deviation for Y translations (100 simulations). 74
5.35 6 DOF simulations: Error and standard deviation for Z translations (100 simulations). 74
5.36 Evolution of the condition number of for one tile with simulated data. 75
5.37 Error and standard deviation for rotations around the X axis using 3D reconstruction (100 simulations). 76
5.38 Error and standard deviation for rotations around the Y axis using 3D reconstruction (100 simulations). 77
5.39 Error and standard deviation for rotations around the Z axis using 3D reconstruction (100 simulations). 77
5.40 Error and standard deviation for X translations using 3D reconstruction (100 simulations). 78
5.41 Error and standard deviation for Y translations using 3D reconstruction (100 simulations). 78
5.42 Error and standard deviation for Z translations using 3D reconstruction (100 simulations). 79
5.43 6DOF: Rotations around X. 81
5.44 6DOF: Rotations around Y. 82
5.45 6DOF: Rotations around Z. 83
5.46 6DOF: Translations along the X axis. 84
5.47 6DOF: Translations along the Y axis. 85
5.48 6DOF: Translations along the Z axis. 86
5.49 Update map for 6DOF. The intensity of a pixel reflects the amount of updates. . . . 87
5.50 Error between the motion estimation of the visuomotor function and the odometry. 87
5.51 Flow chart for processing and matching SIFT keypoints. 88
5.52 SIFT matches. 89
5.53 Trajectory of the robot. 89

A.1 Comparaison between approximations of θ_1 and analytical values. Distribution of relative absolute errors. 105
A.2 Comparaison between approximations of θ_2 and analytical values. Distribution of relative absolute errors. 106
A.3 Comparaison between approximations of θ_3 and analytical values. Distribution of relative absolute errors. 106

A.4 Comparaison between approximations of θ_7 and analytical values. Distribution of relative absolute errors. 107
A.5 Comparaison between approximations of θ_8 and analytical values. Distribution of relative absolute errors. 107
A.6 Comparaison between approximations of θ_9 and analytical values. Distribution of relative absolute errors. 108
A.7 Comparaison between approximations of θ_{14} and analytical values. Distribution of relative absolute errors. 108
A.8 Comparaison between approximations of θ_{15} and analytical values. Distribution of relative absolute errors. 109
A.9 Comparaison between approximations of θ_{16} and analytical values. Distribution of relative absolute errors. 109
A.10 Comparaison between approximations of θ_{20} and analytical values. Distribution of relative absolute errors. 110
A.11 Comparaison between approximations of θ_{21} and analytical values. Distribution of relative absolute errors. 110
A.12 Comparaison between approximations of θ_{22} and analytical values. Distribution of relative absolute errors. 111
A.13 Comparaison between approximations of θ_{27} and analytical values. Distribution of relative absolute errors. 111
A.14 Comparaison between approximations of θ_{28} and analytical values. Distribution of relative absolute errors. 112
A.15 Comparaison between approximations of θ_{29} and analytical values. Distribution of relative absolute errors. 112
A.16 Comparaison between approximations of θ_{33} and analytical values. Distribution of relative absolute errors. 113
A.17 Comparaison between approximations of θ_{34} and analytical values. Distribution of relative absolute errors. 113
A.18 Comparaison between approximations of θ_{35} and analytical values. Distribution of relative absolute errors. 114

List of Symbols

${}^{W}\mathbf{P}$:	homogeneous coordinates of a 3D point in the coordinate frame W
${}^{L}\mathbf{p}$:	projective coordinates of a 3D point in the coordinate frame L
${}^{L}\mathbf{p}_i$:	image point in the left camera
$\mathbf{e}$:	vector of image-based errors
${}^{S}E_W$:	homogeneous representation of a special Euclidean transformation that maps points from the coordinate frame W to the coordinate frame S
${}^{S}\mathbf{E}_W$:	12×1 vector composed of the elements of ${}^{S}E_W$
R:	rotation matrix ($R \in SO(3)$)
$\mathbf{t}$:	3D translation vector ($\mathbf{t} \in \mathbb{R}^3$)
${}^{L}\mathbb{P}$:	set of 3D points that are visible from the left camera (${}^{L}\mathbb{P} \subset \mathbb{R}^3$)
${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$:	set of 3D points that are visible from both stereo cameras simultaneously
$\mathbb{S}$:	set of stereo image coordinates ($\mathbb{S} \subset \mathbb{Z}^4$)
$\mathbb{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$:	set of stereo image coordinates obtained from the projection of points in ${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$
$\mathcal{S}$:	stereo encoding function $\mathcal{S} : {}^{L}\mathbb{P} \cap {}^{R}\mathbb{P} \rightarrow \mathbb{S}$
$\mathbb{s}$:	element of $\mathbb{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$
${}^{L}\boldsymbol{\theta}$:	visuomotor parameters for the left camera (${}^{L}\boldsymbol{\theta} \in \mathbb{R}^n$)
Θ:	function that maps 3D coordinates to a vector of visuomotor parameters ($\Theta : \mathbb{R}^3 \rightarrow \mathbb{R}^n$)
$\Theta_{\mathcal{S}}$:	function that maps stereo coordinates to a vector of visuomotor parameters ($\Theta : \mathbb{S} \rightarrow \mathbb{R}^n$)

Chapter 1

Introduction

An important open research problem in robotics is to control the motion of a robot by using visual feedback. Formally, the definition of vision guided robotics is: Given a task that is derived from images and given that visual feedback is provided by cameras, coordinate and control the motion of the robot to do the task?

Because of its positive repercussion on productivity, industrial robotics is a major economical engine. According to the Robotic Industries Association, the industrys trade group, orders for robots by North American manufacturers surpassed $1 billion in 2005 [13]. Yet, despite decades of intense research and development in robotics, most robots are confined to factories where they do repetitive tasks in well engineered environments such as assembly lines. For example, fixtures are used to carry parts along assembly lines and pegs are used to position and orient the fixtures in the workspace. The accuracy of the fixture mechanisms enables a robot to "blindly" manipulate the content with little or no feedback from sensors. The benefits of these assembly lines are such that they often outweigh the costs of reconfiguring the plants for new products.

Robots, however, are no longer the exclusivity of plants. During the last decade, service robots have emerged in households and in research labs around the world. These robots operate in less engineered environments such as warehouses, houses, on land, underwater and in the air, with little or no human oversight. For example: the Staples warehouse in Chambersburg, PA is automated with storage and retrieval mobile robots, the Roomba vacuuming robot is in more than 1.5 million houses and five competitors finished the 212km course of the 2005 DARPA Grand Challenge. To operate, these robots must sense and interact with their environment. Without arguing about the degree of autonomy of such robots, it is undeniable that they exhibit greater autonomy when compared to their industrial counterparts. Although complete autonomy is a formidable challenge that will seemingly require several more decades to address, low level interaction is within the reach of today's technology. Research in this area mainly focuses on motion planning and control by relying on the feedback from various sensors.

Despite all the variables that are considered to determine which sensor is appropriate for a specific task, a popular solution is to formulate and control tasks from visual information. Irrespective of the motivations, the use of visual feedback to control motion, also known as *visual servoing*, has been consistently at the forefront of research in robotics for over a decade. Since its early days in the mid 80s, visual servoing tracks are now a major component of all major robotics conferences.

The objective of visual servoing is to enable a robot to do tasks by solely relying on visual feedback. First, a task is defined from images and then visual feedback is used to guide the robot toward the goal. In their simplest forms, tasks are defined by the parameters of image features such as the pixel coordinates of a target, the parameters of lines or moments of surfaces. In a more complex form, the tasks are defined by the relative transformations between bodies in the environment.

As with any control mechanism, an important consideration is the convergence of the control law. For some applications, a locally convergent control law is sufficient. In particular, such controllers are applicable to robots working in structured environments [6]. Often, however, it is essential to consider global convergence. Equally important is that many systems must generate an explicit plan on how to do a task. Examples of such systems are industrial robots that only accept position commands or autonomous robots that must be accountable for their actions. Therefore, it is not enough to guarantee that a task will be accomplished as many systems must plan how a task will be accomplished. Despite two decades of intense research in visual servoing, no solution that addresses simultaneously these considerations has been proposed. Global convergence for arbitrary tasks remains an open problem and existing visual servoing methods require assumptions about the task, the environment or the robot. Furthermore, the integration of motion planing with visual servoing has been limited to very specific cases [115].

1.1 Motivation and Purpose

In the literature, the problem of visual servoing is addressed from two perspectives: position-based visual servoing (PBVS) and image-based visual servoing (IBVS) [100, 12, 35, 36]. In PBVS, a task is defined by a Cartesian transformation that represents the relative pose between the coordinate frame of the camera and the coordinate frame of the environment or a body present in the environment. Images are used to estimate the current relative pose between the coordinate frames and the control loop is closed by computing the position error $\mathbf{e}_P$ between the two frames. The error is then related to the camera velocity $\mathbf{v}$ by the position-based interaction matrix L_P according to

$$\dot{\mathbf{e}}_P = L_P \mathbf{v}.$$

The main challenge of PBVS concerns the estimation of pose error $\mathbf{e}_P$. Typically, solutions to this problem require a calibrated camera and knowledge about the environment such as maps for localization [63] or a CAD model to estimate all the degrees of freedom (DOF) of the body [122].

A position-based feed-forward architecture, known as *look-then-move*, consists of estimating the relative pose at sparse way points along a path. In this particular case, visual feedback is only used at each way point and each motion segment consists of a trajectory determined by an independent motion controller. Look-then-move is one of the few strategies that enable feed-forward control because the trajectory during each motion segment is not driven by visual feedback. As a result, look-then-move is the main vision-guided solution for industrial robot because these controllers impose their own trajectory generators.

In IBVS, a task is defined by a vector $\mathbf{p}^*$ representing the desired parameters of a set of image features. The features, such as the coordinates of a target, lines, areas and other moments, are extracted from the images. As the robot moves, each feature is tracked between frames and the current parameters are represented by the vector $\mathbf{p}$. The role of the controller is to regulate each feature to its desired parameters such that the image-based error $\mathbf{e}_I = \mathbf{p}^* - \mathbf{p} = 0$. In the literature, this task is expressed by the first order approximation [64]

$$\dot{\mathbf{e}}_I = L_I \mathbf{v}, \tag{1.1}$$

where L_I is referred as the image-based *interaction matrix*. The idea is to estimate the velocity of the camera at each iteration by measuring an error vector $\mathbf{e}_I$ and solving Equation 1.1. In general, because of the relative ease and robustness of estimating the parameters of image features, the IBVS formulation is known to be computationally cheaper than PBVS. Despite its strengths, the formulation of Equation 1.1 has several theoretical and practical limitations. The main theoretical limitations are its local convergence and the conditioning of the interaction matrix. First, Equation 1.1 represents the first order approximation of image projection [181]. It follows that methods based on Equation 1.1 are asymptotically convergent around $\mathbf{e}_I = 0$ and, thus, more suited for active tracking [43, 135, 130]. Second, the interaction matrix must remain full rank throughout the task. To this day, research in IBVS focuses almost exclusively on answering these two challenges.

The practical limitations of IBVS are particularly acute within the context of autonomous and industrial robotics. First, the control law expressed by Equation 1.1 imposes a trajectory to each image feature, and, consequently, to the robot. This potentially leads to undesirable trajectories [32] or to singularities. Methods for generating image-based trajectories have been developed [141, 165], but such methods do not consider the motion of the robot in Cartesian space. Second, in order to respect the first order approximation, each feature must be tracked at a sufficiently high rate and must remain within the field of view of the camera [45]. Within the context of autonomous robotics, however, requiring the visibility of the features at all time imposes several practical limitations. First, it is possible to imagine a situation in which an autonomous robot must do a task that involves moving behind an obstacle and loosing visual contact with the targets. Second, because IBVS forces the robot to keep all targets within the field of view of the camera, IBVS tasks impose important constraints on the motor system especially for tasks that involve six degrees of freedom. This limits the performance of other concurrent tasks that share the vision or the motor systems. Examples of such tasks includes: panning to detect obstacles and map the environment, shift of attention, and responding to other perceptual cues such as noises. Likewise, typical industrial manipulators can only interpolate their motion in joint or Cartesian space and these trajectories can push features outside the field of view [114, 115].

In summary, despite two decades of intense research in the field of visual servoing, no satisfactory solution has addressed the challenge of image-based feed -forward control faced by autonomous robots or industrial robots. Interestingly, PBVS and IBVS complement their respective strengths. On one hand, PBVS is globally convergent and, in particular, look-then-move is suitable for feed-forward control and motion planning. This is a virtue for autonomous and industrial robots because these robots must be able to plan each motion ahead of execution. The challenge of pose estimation, however, makes PBVS and look-then-move less attractive for non-engineered environments. On

	Feedback	Feed-forward
Image-Based	IBVS Input: Image-based velocity $\dot{\mathbf{e}}_I$ Output: Velocity $\mathbf{v}$ Control Law: $\dot{\mathbf{e}}_I = L_I\mathbf{v}$	Visuomotor Function Input: Image-based error $\mathbf{e}_I$ Output: Position $\mathbf{E}$ Control Law: $\mathbf{e}_I = V\mathbf{E}$
Position-Based	PBVS Input: Position-based velocity $\dot{\mathbf{e}}_P$ Output: Velocity $\mathbf{v}$ Control Law: $\dot{\mathbf{e}}_P = L_P\mathbf{v}$	Look-then-move Input: Position-based error $\mathbf{e}_P$ Output: Positions $\mathbf{E}$ Control Law: $\mathbf{e}_P = I\mathbf{E}$

Table 1.1: Compared to other methods, the visuomotor function defines a control law that maps image-based errors to position commands.

the other hand, IBVS is locally convergent and does not extend to feed-forward control and motion planning.

The purpose of this thesis is to propose the *visuomotor function* as a new solution to feed-forward control of vision-guided robots. The proposed system merges the benefits of look-then-move with those of IBVS. The visuomotor function provides the capability of feed-forward control in non-engineered environments by using image-based errors. An important aspect of this research is that it does not propose a switching control method where the position-based controller and the image-based controller are used in alternation [57]. Instead, the proposed method truly merges both methods by using the range of IBVS control laws and the domain of look-then-move control laws. The comparison between the proposed research and the existing methods is illustrated in Table 1.1. Table 1.1 shows the combinations of feed-forward and feedback control loops versus image-based and position-based errors. While researchers have mostly focussed on IBVS and PBVS, the visuomotor function is the only known method that aims to map directly image-based error to position commands in different task spaces.

1.2 Objectives

Human performance is used as the gold standard in many areas of robotics and computational intelligence [50]. Hand-eye coordination is no exception to this rule as human visuomotor performances are arguably the most developed. Although the visuomotor system in humans is too complex and misunderstood to be considered as a benchmark, its capabilities and limitations are well defined and are relevant to outline the specifications of visuomotor systems for autonomous robots.

An early theory about the human visuomotor system was based on a construction of a 3D representation of the environment from images [137]. As a first stage, the visual input is processed to recreate the 3D geometry of the environment that surrounds an individual. In the second stage, the 3D representation is used by the motor system to plan motion and control the movement. Later, this theory was undermined by neurophysiological research that demonstrated the existence of the dorsal and ventral streams in the brain and their distinct role regarding motion control and perception [143]. The role of the dorsal stream for hand-eye coordination was demonstrated by noting that individuals with perceptual deficiencies are able to control their motion from visual input. For example, patients with visual agnosia[1] preserve their visuomotor abilities and are able to accomplish vision guided tasks even though their perception of the tasks is inconsistent with reality [143]. Although counter-intuitive, such findings suggest that humans are able to do a task despite being able to perceive their environment correctly.

Also, it was found that the hand-eye system of humans is well calibrated and is capable of feed-forward motion [140]. For example, experiments have demonstrated the ability of humans to perform visuomotor tasks without using visual feedback by reaching for illuminated targets in darkness [82].

Finally, the capability of humans to recalibrate their visuomotor system is well documented. Experiments have demonstrated this by having subjects performing a task while distorting the visual input with lenses or prisms [40]. Results showed that humans are able to compensate for the visual distortion and to recalibrate their visuomotor system. It appears, however, that the visual system is not significantly affected by the recalibration. Instead, alteration of the proprioceptive representation

[1]Patient with visual form agnosia have visual perception deficiencies. For example, they fail to recognize or replicate objects or geometrie despite being able to detect them.

accounts for most of the recalibration [185, 17]. Also, the recalibration affects only the limbs that are observed under the visual distortion, which suggest that the recalibration does not even affect the whole visuomotor system [38, 19].

These studies demonstrate that vision-guided motion control and planning in human does not depend on 3D perception of the environment, but rather on a dedicated pathway that maps visual input to motor output [143]. Without claiming to replicate the human visuomotor system, these aforementioned characteristics are the inspiration behind the research presented in this thesis. The objective of this thesis is to combine such capabilities in a single vision guided framework called the visuomotor function. The objective of the function is to map image-based errors directly to special Euclidean transformations without requiring the knowledge of 3D geometry or pose estimation.

Following the discussion about the capabilities of human's visuomotor system, the objective of this thesis is to propose a visuomotor function with the following characteristics:

1. The visual input must be image-based and only the coordinates of image targets are extracted from the images.
2. The output of the control law is a transformation in a task space (i.e. $SE(3)$). That is, path planners and trajectory generators can be used to determine how the robot do a task.
3. The visuomotor is an *interaction matrix* that maps directly the input to the output.
4. Knowledge or the estimation of the structure of the environment, such as the 3D geometry, is not required.
5. The approximation of the visuomotor function must be transparent and on-line, that is the visuomotor system must self-calibrate.

1.3 Contributions

Based on the discussion of sections 1.1 and 1.2, this thesis introduces the visuomotor function. The function relates arbitrary errors in the image space to a motion of the camera in The integrity of the visuomotor function is ensured by a transparent and continuous approximation algorithm.

Compared to IBVS that relates image-based errors to the velocity of the camera, the visuomotor function defines an interaction matrix that maps the same error to a transformation in the task space of the robot (i.e. $SE(3)$).

The merit of the visuomotor function is that it addresses challenges that are specific to autonomous and industrial robots. First, because the range of the visuomotor function represents position commands, it enables the use of path planners and high level decision making algorithms. That is, since the solution to the visuomotor function is a rigid transformation that is expressed with respect to the current position of the end-effector, a trajectory generator can outline a path in the task space between the current and desired position of the end-effector. Similarly, this transformation can be passed to a path planner such as a PRM [106] to avoid collisions and other obstacles. Second, because the errors are image-based, the system can be used easily in non-engineered environments.

The research included in this thesis has been the focus of the several scientific contributions. The contributions are presented in chronological order.

- [116, 117, 118] These contributions introduced the idea of relating arbitrary image-based variations to variations in the configuration space of the robot. Even though the mathematical formulation of the visuomotor function is specific to a pan-tilt unit, the papers present the advantages of the visuomotor function for the problem of controlling a robotic head. The papers present techniques and algorithms used in the subsequent papers. In particular, recursive least squares [118] is used to estimate the parameters of the visuomotor function and cerebellar model articulation controller [116, 117] is used to generalize them and approximate the global function.
- [119] This contribution introduced the derivation of the visuomotor function for translational tasks. The theory presents how the visuomotor function relates the coordinate errors of image targets to motor error in translational task space. Using the method of recursive least squares presented in the previous papers, the parameters of the function are estimated on-line and the system is able to develop hand-eye coordination for reaching tasks in a short time.
- [120] This paper extends the formulation of the visuomotor function presented in [119] to the displacement of all six degrees of freedom (translations and rotations) of the end-effector. The main contribution is the derivation of the linear equations that relate the error in feature space to the error in the special Euclidean space ($SE(3)$). Despite the greater dimensionality of

the new formulation, experiments showed the convergence of the system in a relatively short time.

- The visuomotor was also derived for the task space of mobile robots. In this research, the task space consists of planar translations and panning. The result is a compact representation of the visuomotor function.

Chapter 2

Background and Review

2.1 Visual Servoing Architectures

The earliest use of visual feedback in robotics is traced back to the 1960s [50]. Within the structured environment of the Blocks Micro World, Marvin Minsky and Seymour Papert used a robotic arm with five degrees of freedom and a vision system to construct structures with blocks. Their program was derived from earlier work by Patrick Winston in which a computer learned to recognize simple block scenes. Later, their research spun off the Copy Demo project led by Winston and Berthold Horn [187]. In the late 1960s, a similar project assembled car water pumps from randomly scattered parts [23].

Throughout the following decades, robot vision emerged as a field of its own. Among the areas that compose robot vision, visual servoing addresses a fundamental problem of how to use visual feedback to control the motion of a robot [12, 100, 35, 36]. Visual servoing systems are defined by the domain and the range of control law. In general, the input space is derived from the visual system whereas the output space aims at the motor system. Within these guidelines, many variants have been investigated. According to the literature, input spaces are either labeled as *position-based* or *image-based* with some specific hybrid cases. Typically, the ranges of the control laws are referred as *direct* or *indirect*.

2.1.1 Input: Visual System

In visual servoing, the domain of the control law is used to define the commands and, consequently, to determine the processing involved by the visual system. Informally, any information extracted can be used to specify a command and this diversity is reflected in the literature with commands defined by: points, lines, image moments, poses, optical flow, triangular meshes, homographies and luminance. Because every domain has different attributes, the choice of a specific domain, or a combination thereof, depends on the application. In some applications, it is possible to estimate the relative pose between the coordinate frame of a body and the coordinate frame of the camera. Under these conditions, a command represents the desired relative pose between the coordinate frames of the body and the camera. In general, this domain corresponds to the special Euclidean $SE(3)$ group of transformations, but it is possible to restrict the domain to a specific subspace.

If the pose cannot be estimated, then the parameters of image features can be used as input. Obviously, the input space varies according to the type and the number of features that are used. For these reasons, it is convenient to represent the parameters of all features by a d dimensional vector. An *image-based* command in $\mathbf{p}^\star \in \mathbb{R}^d$ specifies the desired parameters of each selected features. The most common features are: points, lines and surface moments (area, centroid and covariance).

2.1.2 Output: Motor System

The range of the control law is defined by any space that can be used to drive a motor system or any related sub-controller. In general, these spaces represent any positions, velocities and accelerations that are defined in either joint space or operation space. In these cases, known as *indirect visual servoing*, the range of the control law must be fed to sub-controllers that are responsible for driving the motor system.

The range can also span forces and moments in operation space or torques in joint space. In the literature, these controllers are referred as *direct visual servoing* because the error is used to drive

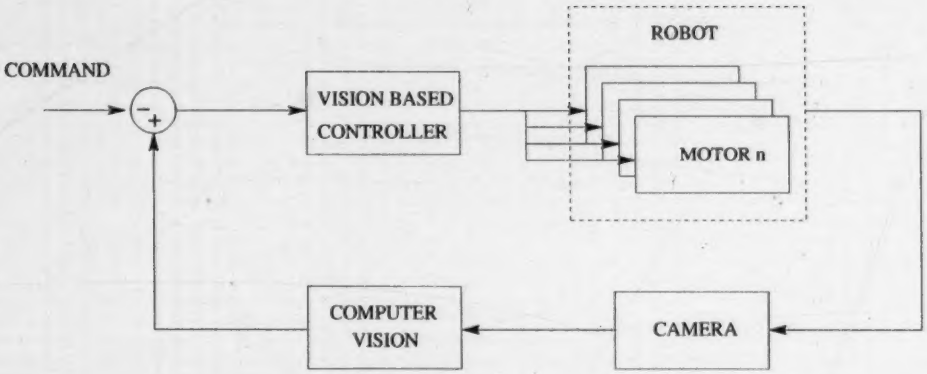


Figure 2.1: Direct visual servoing.

each motor di[illegible]. Direct visual servoing generally involves the additional burden of the dynamics of the manipul[illegible] [145].

2.2 Direct visual servoing

In direct visual servoing, the control law directly maps the input error to the motors as torques as illustrated in Figure 2.1.

Research in direct visual servoing has been at best fragmentary for several reasons. First, the frequencies required to control the motor system of a manipulator are typically much higher than the rates of commercial cameras. For examples, the frequency of Unimation Mark II controllers operate at 1kHz [47] and the controller of the Whole Arm Manipulator (WAM) used in this thesis operates at 500Hz. In comparison, most cameras operate at 30 frames per seconds and few advanced cameras have rates as high as 200 frames per second and experimental imaging systems have rates up to 1000 frames per second [146]. Low frequency is thus a main concern in the stability analysis of a direct visual servoing system [45]. Moreover, high frame rates increases the bandwidth requirements and the toll on the computing resources to process each frame. Thus, either the computing resources must be scaled for the additional image processing or computationally inexpensive image processing algorithms must be considered. Second, many robotic platforms accept Cartesian position commands. With this convenient abstraction, the end-effector of a manipulator is represented as a coordinate frame in Cartesian space. This representation greatly simplifies the design of algorithms and increases the portability between robots.

Nevertheless, some research groups have concentrated their efforts on direct visual servoing. Most notably, Kelly showed how a two links planar robot can be controlled from visual feedback [107]. The task consists of moving the end-effector over a target by using a camera placed above and perpendicular to the workspace of the robot. The image coordinates of the target and the end-effector are measured in the image and the error is the difference between the two measurements. The controller outputs the torque of each motor from the evaluation of the manipulator's Jacobian. This paper exposes the complexity created by the inclusion of the manipulator's dynamics in the control law. Furthermore, the experiments consist of simulations and assume continuous visual feedback. A similar controller was presented in [108]. This paper introduced a saturation function in the control law to limit joint torques and, consequently, large motion between images.

Direct visual controllers are evaluated in [160]. The aforementioned controllers [107, 108] are compared to a controller similar to the one presented in [160]. Again, the experiments are based on the simulation of a planar manipulator with two degrees of freedom. Results indicate that the transient response of the controller proposed in [107] is the shortest but also suffers from overshoot. In comparison, the controller presented in [108] does not overshoot, but has a longer response.

A recurrent problem with direct visual servoing is the instabilities caused by the latency and sampling rate of the vision system [45, 183]. This problem becomes chronic when a target is maneuvering. To overcome this problem, Koivo and Houshangi developed an adaptive controller based on a auto-regressive model to grasp a moving target with unknown dynamic [111].

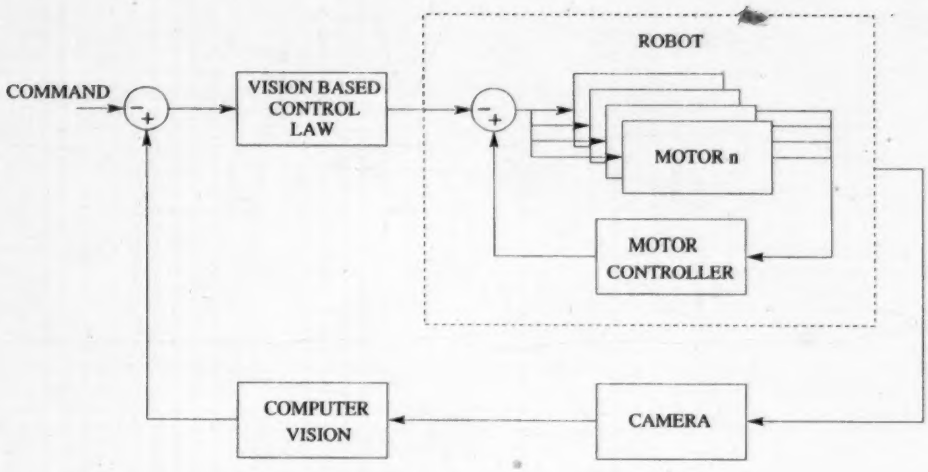


Figure 2.2: Direct visual servoing.

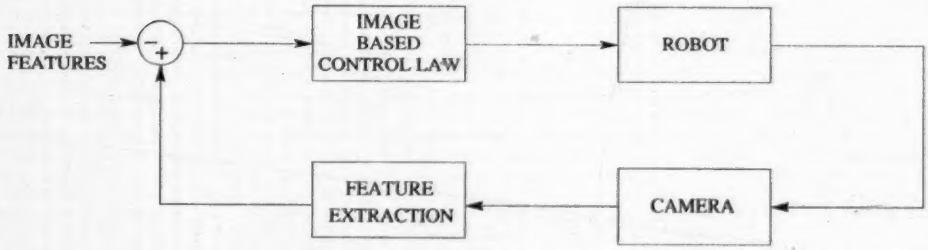


Figure 2.3: Image-based visual servoing

2.3 Indirect Visual Servoing

Because of the complexity of direct control laws, indirect controllers have become the *de facto* benchmark in visual servoing. An indirect control law outputs commands that are expressed in Cartesian space. Typically, these commands are defined by transformation in the space of special Euclidean transformations ($SE(3)$) or twists. Then, these commands are passed to lower level controllers that regulate each motor of the system. The indirect architecture is depicted in Figure 2.2.

Typically, the inside loop is composed of joint controllers that operate at a suitable rate[45]. This architecture allows the visual feedback to operate at a much lower rate because it is confined to the outer loop. As the large majority of the research in visual servoing is based on the indirect architecture, the review of indirect visual servoing methods will follow in the upcoming sections. The following section will classify the control laws according to their domain. Namely, if the control laws are position-based or image-based.

2.4 Image-Based Visual Servoing

Image-based visual servoing consists of performing a "visual alignment" by directly regulating the parameters of image feature. That is, the command specifies the desired parameters of a set of selected image features. As illustrated in Figure 2.3, image-based visual servoing methods propose control laws that regulate the parameters of image feature to their desired values.

Virtually any image feature can be used for image-based visual servoing. However, the most popular feature, and perhaps the most intuitive for understanding IBVS, are image points. An image point $\mathbf{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$ has two parameters that represent the horizontal and vertical image coordinates of a three dimensional target $\mathbf{P}$. A command, denoted $\mathbf{p}_i^*$, represents the desired image

coordinates for $\mathbf{P}$. Given $\mathbf{p}_i^*$ and $\mathbf{p}_i$, the role of the controller is to move the robot such that the error $\mathbf{e}_i = \mathbf{p}_i^* - \mathbf{p}_i = \mathbf{0}$.

In IBVS, the role of the vision system is limited to extracting features from the images. Since feature extraction does not rely on calibrated cameras, IBVS has little or no bias and the task is completed when each feature $\mathbf{e} = \mathbf{0}$. This characteristic is known to make IBVS methods more robust than their PBVS counterparts. Another advantage is that a command can be directly extracted from an image. That is, a command can be generated by showing the desired image to the system and extracting the desired features from the image.

The main challenge of IBVS lies in the design of the control law or selecting the appropriate features for a given task. An early IBVS method using image points was introduced in [184]. In addition to introducing most of the IBVS terminology, the authors are among the first to formulate the problem as a first order approximation. They also provide tools for assessing the stability of their controller. Weiss *et al.* introduced the nature of the relationship between the velocities of image points $\dot{\mathbf{p}}_i$ and the body velocity of the camera's $\mathbf{v} = \begin{bmatrix} \nu & \omega \end{bmatrix}^T$, where ν represents the linear velocity and ω represents the angular velocity.

They expressed this relationship as

$$\dot{\mathbf{p}}_i = L\mathbf{v} \tag{2.1}$$

where L is known as the *interaction matrix*[1]. Their formulation of L, however, lacks the knowledge of a realistic camera model. As such, they were not able to pin down an exact analytical formulation of L.

With progress in computer vision, Espiau *et al* [64] were able to detail the nature of the interaction matrix for several features. First, they outline the nature of the interaction matrix. Then, they formulate the problem of IBVS as a *task function* [164] given by

$$\mathbf{e}_I = \mathbf{p}_i^* - \mathbf{p}_i \tag{2.2}$$

in which the error $\mathbf{e}_I$ must regulate to $\mathbf{e}_I = \mathbf{0}$. Then, they derive an image-based interaction matrix L_I that relates the velocity of the camera $\mathbf{v}$ to the time derivative of the error $\dot{\mathbf{e}}_I$ with

$$\dot{\mathbf{e}}_I = L_I\mathbf{v}. \tag{2.3}$$

Substituting $\dot{\mathbf{e}}_I = -\lambda\mathbf{e}_I$ and solving Equation 2.3 results in

$$\mathbf{v} = -\lambda L_I^+ \mathbf{e}_I \tag{2.4}$$

where L_I^+ represents the Moore-Penrose pseudo-inverse of L_I. The differential nature of equation 2.3 ensures the exponential decay of $\mathbf{e}_I$.

The interaction matrices proposed in [64] are Jacobians that relates the instantaneous variations of image features to the twist of the camera. As an example, the formulation of the interaction matrix for image point given by

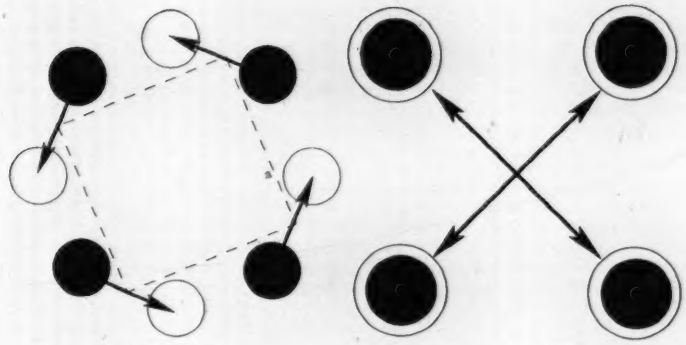
$$L_I(\mathbf{p}_i, {}^C z) = \begin{bmatrix} \frac{f}{{}^C z} & 0 & \frac{-x_i}{{}^C z} & \frac{-x_i y_i}{f} & \frac{f^2+x_i^2}{f} & -y_i \\ 0 & \frac{f}{{}^C z} & \frac{-y_i}{{}^C z} & \frac{-f^2-y_i^2}{f} & \frac{x_i y_i}{f} & x_i \end{bmatrix} \tag{2.5}$$

where f is the focal length of the camera lens. An important observation is that the evaluation of L_I depends on the parameters $\mathbf{p}_i$ and the depth ${}^C z$ of the corresponding 3D point $\mathbf{P}$. In that respect, IBVS requires calibrated cameras and the 3D information of each point. Methods such as [55] can be used if a 3D model of the scene is available. Others have used motion cues [34, 170] to estimate the depth of image points.

Several problems arise from the formulation of Equation (2.4). First, poor conditioning and singularities of L_I results in instabilities. Although methods to avoid singularities have been proposed [167], such methods assume that the task does not constraint all the degrees of freedom and the singularities are avoided by motion within the kernel of L_I.

Second, the exponential decay of $\mathbf{e}_I$ dictates the trajectory of $\mathbf{v}$ and thus the motion of the robot. By Equation 2.4, the control law clearly regulates the error of the features parameters. Under this control law, each feature follows an image-based trajectory that drives the exponential decrease of $\mathbf{e}_I$. Although these trajectories are favorable in the image space, the corresponding trajectories in the configuration space of the robot are not always desirable. A classic example of this, known as the *Chaumette conundrum* [32], is illustrated in Figure 2.4(a). The initial coordinates of four points are denoted by black dots. The task consists of moving the points to their desired coordinates indicated

[1] The terms "image Jacobian" and "interaction matrix" are often used interchangeably in the literature.



(a) 45° rotation. (b) 180° rotation.

Figure 2.4: Chaumette conundrum.

by circles. Clearly, the only motion required in this example is to rotate the camera around its optical axis by 45°. Under the image-based control law of equation 2.4, however, each point describes a straight line between its initial and final coordinates as indicated by the arrows. If a snapshot is taken during this trajectory, the square enclosing the four points (dashed square in Figure 2.4(a)) is smaller than the original enclosing square. This suggest that the camera needs moves backward while rotating. An extreme instance of this problem is illustrated in Figure 2.4(b) where a pure rotation of 180° around the optical axis is desired. To follow this image-based trajectory, the camera first moves back at infinity such that all the points intersect in the center, then rotate 180° and, finally, move forward.

Third, the differential nature of equation (2.4) also suggests that it is only locally valid. As such, the equation is only valid around the neighborhood of $\mathbf{p}_i$. For certain configuration of image features and $\mathbf{p}_i - \mathbf{p}_i^*$, the system may converge to a local minimum [32]. This will be analyzed in further details in the coming sections.

Finally, the exponential decrease of $\mathbf{e}_I$ typically implies that the initial camera velocities $\mathbf{v}$ is violent and decreases as $\mathbf{e}_I$ decreases. This characteristic further complicates motion planning as the gain λ of the controller must be adapted to decelerate or accelerate the motion of the robot.

Overcoming these challenges has been the topic of most visual servoing research during the last decade. A strategy is to improve the control laws by avoiding direct evaluation of the interaction matrix or by modifying the nature of equation (2.4). Another strategy is to design image features with desirable properties for specific control tasks.

2.4.1 Control Laws

Most of the research in IBVS focuses on addressing the challenges related to the evaluation of the interaction matrix. The common denominator is to replace the exact evaluation of L_I^+ by an approximation $\widehat{L_I^+}$. These solutions are grouped according to the estimation method. Others, known as hybrid controllers, aim to decouple L_I^+ to obtain desired behavior.

Time Varying Interaction Matrix

Instead of evaluating L_I^+ at each iteration, the interaction matrix is incrementally updated by using the input/output variations observed between consecutive iterations. In [103] the update algorithm is based on Broyden's method for secant updates. Given an initial approximation, the interaction matrix incorporates the previous variations into the current estimate of L_I^+. Formally, the expression for L_I^+ at the $k+1$ iteration is obtained from L_I^+ at the k iteration according to

$$\widehat{L_I^+}(k+1) = \widehat{L_I^+}(k) + dK.$$

The main advantage of this method is the approximation of the interaction matrix without knowledge of f and ${}^C z$.

Depth Invariance

In this method, the depth ${}^{C}z$ of each point is set to the depth ${}^{C}z^{*}$ at its desired location. Thus, $\widehat{L_I^+} = L_I^{+*}$ is constant and only the depth of each point in the desired pose is required. Convergence is only guaranteed in the neighborhood around the desired configuration. In addition, the constant interaction matrix can lead to undesirable motion and features are not constrained to remain within the field of view.

Projective Invariance

Projective invariance is used to design interaction matrices that do not depend on the intrinsic parameters of the camera [127]. The method define a projective space in P^2 by assigning the projective basis to three non-collinear points in the image and projecting the remaining points on the basis. The result is the collapse of the interaction matrix to the identity matrix, such that $\widehat{L_I^+} = I$. The analysis of local stability for large calibration errors is reported in [128].

2nd Order Approximation

Instead of the first order approximation of Equation, a second order approximation is obtained by formulating L_I as [129]

$$\widehat{L_I} = \frac{1}{2}(L_I + L_I{}^{*}) \tag{2.6}$$

where $\widehat{L_I}$ represents the average of the current interaction matrix L_I and the interaction matrix in the desired configuration $L_I{}^{*}$. The presence of L_I in Equation 2.6, however, suggest that knowledge of the depth of each point is required.

Hybrid IBVS

Hybrid control laws decouple a task or its degrees of freedom in two or more independent control laws. The idea is to control each component individually.

Perhaps the most cited example of this approach is the 2½ method presented in [133]. The stability of the method is presented in [132]. 2½ visual servoing controls the motion by using two decoupled control processes simultaneously. The first one controls the rotational component and the second one controls the translational component. It follows that the right hand side of equation (2.4) is written as

$$L\mathbf{v} = L_\nu(\mathbf{p}_i, Z)\boldsymbol{\nu} + L_\omega(\mathbf{p}_i)\boldsymbol{\omega} \tag{2.7}$$

where $L_\nu(\mathbf{p}_i, Z)$ and $L_\omega(\mathbf{p}_i)$ are the columns of the Jacobian corresponding to translational and rotational motion.

The decoupling of the translation and the rotation are explained as follow. Given a planar surface, it is possible to estimate the rotation between two views from the corresponding homography. If a 3D point $\mathbf{P}$ lies on a plane and its projective coordinates are given by $\mathbf{p}$, then the transformation of the plane by an homogeneous transformation

$$E = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.8}$$

with $(R, \mathbf{t}) \in SE(3)$, correspond to a 3×3 homography H of the image plane. That is, given

$$\mathbf{P}' = E \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix}$$

we obtain [67]

$$\alpha \mathbf{p}' = H\mathbf{p}$$

where α is a scale factor and $\mathbf{p}$ and $\mathbf{p}'$ are the projective coordinates of $\mathbf{P}$ and $\mathbf{P}'$ respectively. Since H is defined up to a scale factor it has 8 degrees of freedom and a solution requires four pairs of corresponding points $\mathbf{p}' \leftrightarrow \mathbf{p}$. Using H, the rotation R and a scaled translation are recovered with [67, 131]

$$H = R + \frac{\mathbf{t}^T \mathbf{n}}{d^*} \tag{2.9}$$

where $\mathbf{n}$ is the normal of the planar surface and d^* is the distance between the camera at the desired location and the planar surface. Following this, the rotation is controlled with PBVS (3D) while the translation is controlled under IBVS (2D).

Another similar approach was suggested in [53]. Instead of extracting the rotation R, the translational motion is first handled by

$$\mathbf{v} = \mathbf{t}\frac{d}{d^*}$$

where d is the distance between the camera in its current position and the plane. Using this, the rotational motion is controlled by

$$\boldsymbol{\omega} = -L_\omega^+(\dot{\mathbf{p}} + L_\nu \boldsymbol{\nu}).$$

Recently, Corke addressed the camera retreat problem (Figure 2.4(a)) by controlling the z axis independently from the x and y axes [46]. The control law is expressed as

$$\dot{\mathbf{p}} = L_{xy}\dot{\mathbf{r}}_{xy} + L_z\dot{\mathbf{r}}_z$$

where $\mathbf{r}_{xy} = \begin{bmatrix}\nu_x & \nu_y & \omega_x & \omega_y\end{bmatrix}^T$ and $\mathbf{r}_z = \begin{bmatrix}\nu_z & \omega_z\end{bmatrix}^T$. To enable this partitioning, the method relies on two novel image features. The first one, θ_{ij}, controls the rotation around the optical axis. It is determined by the angle between the line connecting the points $\mathbf{p}_i$ to $\mathbf{p}_j$ and the horizontal baseline. The control of the rotation around the optical axis (ω_z) becomes

$$\omega_z = \gamma_{\omega_z}(\theta_{ij}^* - \theta_{ij})$$

where γ_{ω_z} is the control gain. The second feature, σ, represents the area of the polygon determined by the points. It is used to control translations along the optical axis and is invariant to rotation about the same axis. Control of this feature is given by

$$v_z = \gamma_{v_z}(\sigma^* - \sigma). \tag{2.10}$$

A comparison of previous three hybrid methods ([133, 53, 46]) is reported in [75].

In [126], each point of the target is projected onto a spherical image of unit radius and the feature is the centroid of the projections. Using the velocity of the centroid in the image enables to decouple the translational and rotational velocities.

Instead of decoupling the velocity of the end effector, Oh and Allen partition the degrees of freedoms of the robot [148]. Their method allows the specialization of each DOF according to their transient response. Joints with large bandwidth are used for fast moving targets and the others are used for slow targets.

Other Control Law

Classic control laws (PI, linear quadratic Gaussian and pole assignment) were evaluated for visual servoing in [152]. The feature used is the optical flow of a target moving at a constant depth. The strategy is for the controller to regulate the flow to zero. The same authors analyzes three adaptive controllers in [151] along the same benchmarks.

Ruf *et al.* proposed a different relationship between the velocities of features and the camera velocity [162]. Instead of relating variations in features space to the camera velocity, the velocity is expressed in a projective space such that the interaction matrix relates two projective spaces. That is, rotations and translations of the camera are represented by projective transformations by deriving a projective interaction matrix.

2.4.2 Stability

The stability of IBVS is analyzed by using the Lyapunov function candidate

$$\mathcal{L}(t) = \frac{1}{2}\|\mathbf{e}_I(t)\|^2$$

with the time derivative

$$\begin{aligned}\dot{\mathcal{L}}(t) &= \mathbf{e}_I^T(t)\dot{\mathbf{e}}_I(t) \\ &= -\lambda \mathbf{e}_I^T(t) L_I \widehat{L_I^+} \mathbf{e}_I(t).\end{aligned}$$

From Lyapunov second theorem, Equation 2.4 is asymptotically stable if $\dot{\mathcal{L}}$ is negative definite. For this it is sufficient to show that $L_I\widehat{L_I^+}$ is positive definite. Because L_I is an $M \times 6$ matrix ($6 \leq M$) and $L_I\widehat{L_I^+}$ is an $M \times M$ matrix with $\text{rank}(L_I\widehat{L_I^+}) \leq 6$, then $(L_I\widehat{L_I^+})$ is at best positive semi-definite. Therefore, IBVS methods are not globally convergent and only asymptotic convergence around $\mathbf{e}_I = \mathbf{0}$ is ensured [35].

2.4.3 Image-Based Trajectories

A drawback of the interaction matrix stems from the representation of a first order approximation of a nonlinear system. The implication of the first order approximation is that it converges asymptotically around $\mathbf{e}_I = \mathbf{0}$. Attempts for global convergence have mainly used image-based trajectories $\mathbf{p}_0, \ldots, \mathbf{p}^*$ to insert intermediate commands between the initial image features $\mathbf{p}_0$ and the desired one $\mathbf{p}^*$. These commands are applied successively to the controller such that each feature follows a deterministic trajectory. Because the 3D geometry of bodies involved in an image-based servoing task is unknown, these trajectories are not planned offline but instead are computed on the fly according to some rules.

Feddema and Mitchell designed a feature-based trajectory generator to compute a sequence of pixel coordinates between a current view and a desired view [68]. Their generator allows asynchronous control and ensures that features stay within the field of view. The trajectory predicts the path of a feature by using velocity constraints of the features within each segment of the trajectory. The feature used in their work is the center point of the segment linking two points, the segment's length as well as its orientation. They also derive a Jacobian in order to predict how the features move between two control steps. Their method, however, can control only one feature at the time because the combined trajectories can violate the rigidity constraint of the body.

Park uses an uncalibrated stereo system to synthesize intermediate views of a gripper [153]. These intermediate images are inserted between the initial and desired configurations to generate intermediate visual servoing goals. Another path planing method interpolates images stored in a tracking database to generate intermediate goals [142]. Similarly, trajectory generation is used for a collision detection algorithm based on visual servoing [98] with uncalibrated stereo vision. As in [39], it uses the constraint that projected obstacles cannot intersect the projected trajectory. Potential fields are used in [141] to generate image-based trajectory where the desired configuration defines the attractive field and the repulsive field is defined by obstacles and image borders. Another method also based on potential fields is presented in [90]. Recently, Schram proposed a trajectory generator that minimizes the amount of translation while keeping the target within the field of view [165].

2.4.4 Camera Model

Like other problems in computer vision, using a specific camera model can often simplify the formulation at the cost of a few assumptions. In [41], the mapping between the initial and reference views is captured by an affine transformation by assuming a planar target and an affine camera model. The velocity of the end-effector is related to the degrees of freedom of the affine transformation by a linear transformation based on the parameters of the plane. A similar method is extended to handle discrete time and adaptive to depth in [44].

The same method is expanded in [5] by modeling the image velocity by a linear transformation around the centroid of the object and formulating an interaction matrix that relates two-dimensional appearance variations to 3D relative motion between the camera and the body. Drummond and Cipolla use a similar approach by approximating the affine transformation by a linear combination of basis functions derived from Lie algebra [62].

Others have focused on increasing the robustness of IBVS. Since the interaction matrix depends on intrinsic parameters of the camera, calibration errors can affect the stability of the system. Control methods that are invariant to intrinsic parameters were presented in [84, 127]

Recently, omnidirectional cameras [79, 80] have been used in IBVS [177, 71]. The idea is to use the projection of features on an omnidirectional camera to decouple the rotation degrees of freedom from the translation degrees of freedom.

2.4.5 Stereo IBVS

A study of tasks that can be accomplished using stereo cameras with and without proper calibration is presented in [96]. The authors elaborate on the conditions under which positioning tasks can be accomplished with IBVS or PBVS. Some of the conclusions are

- Positioning is possible with a weakly calibrated stereo system if the task to be achieved is projective invariant.

- Only specific projective invariant tasks are possible with uncalibrated stereo system.

IBVS can be extended seamlessly to stereo cameras. For stereo cameras, one must note that both cameras do not share the same coordinate frame such that one of them must act as the reference frame. Since, the velocity of the left camera ${}^{L}\mathbf{v}$ relates to the velocity of the right camera ${}^{R}\mathbf{v}$ by ${}^{L}\mathbf{v} = \mathrm{Ad}\,{}^{R}\mathbf{v}$ where Ad is the adjoint matrix defined by [145]

$$\mathrm{Ad} = \begin{bmatrix} R & [\mathbf{t}]_{\times} R \\ 0 & R \end{bmatrix},$$

the interaction matrix for stereo cameras is denoted by

$$L = \begin{bmatrix} {}^{L}L_I \\ \mathrm{Ad}\,{}^{R}L_I \end{bmatrix}. \tag{2.11}$$

An important observation is that although each stereo point provides 4 equations, the rank of Equation 2.11 is 3 because each stereo point (4D) encodes a 3D point.

An early stereo IBVS was introduced in [97]. Their method uses an affine camera model to estimate the geometry. Active contours are used to track the gripper with the constraint that it deforms according to the camera model.

Hager proposed one of the few visual servoing systems with a standalone stereo rig [86]. The image projections of a 3D point $\mathbf{P}$ on the robot's end effector is defined by $\begin{bmatrix} {}^{L}\mathbf{p}_i^T & {}^{R}\mathbf{p}_i^T \end{bmatrix}^T$ where ${}^{L}\mathbf{p}_i^T$ and ${}^{R}\mathbf{p}_i^T$ are the projection of $\mathbf{P}$ in the left and right images respectively. The first task is to project $\mathbf{P} \in \mathbb{R}^4$ into a lower space with a transformation $\Phi : \mathbb{R}^4 \rightarrow \mathbb{R}^3$. This transformation consists of projecting ${}^{L}\mathbf{p}_i^T$ and ${}^{R}\mathbf{p}_i^T$ on one of the epipolar lines and then on the line perpendicular to it. The result is obtained by keeping each projection on the epipole and averaging the projections on the lines perpendicular to it. This work was extended to different alignment tasks in [85] namely: point to point, point to line and line to point.

2.4.6 Image Features

All the IBVS methods discussed so far were developed around point features where the parameter of each feature are (x, y) image coordinates.

One of the few methods to use line features is presented in [6]. The research targets automated welding applications where a manipulator mounted with a welding arc must follow the seam of two intersecting metal sheets. Because the system uses line features, it also implies a form trajectory generation as the tool control point must follow the line. The line is represented by bi-normalized Plücker coordinates, which define a pencil of parallel lines lying on a plane, and the depth of the line. With these coordinates, a switching control strategy is used to decouple the control of rotation and translation the system first orients and then positions the camera.

IBVS has also been used with fields of optical flows, such as the method discussed in [49] and with some applications presented in [48]. The image flow is modeled as a polynomial whose parameters are used as features for IBVS. The location of an image point $\mathbf{p}_i$ after k frames is defined by

$$\mathbf{p} = \mathbf{p}_0 + \sum_{i=1}^{k} \dot{\mathbf{p}} \delta t$$

where δt is the time interval between each image and $\dot{\mathbf{p}}$ is estimated by a quadratic motion model.

Chaumette uses image moments of binary images [33]. The image Jacobian is designed to relate camera velocity to moments variations. To use this method, the object must be segmented from the image and only planar objects are considered.

A quantitative approach to measure how the performances (speed and accuracy) of image-based visual servoing depends on the selection of features is presented in [91]. The measure, called *sensitivity*, is related to the rank of the image Jacobian and consequently to the stability of the system. It is shown that the sensitivity is zero when the image Jacobian is rank deficient and that using redundant features increases the sensitivity and performances.

2.5 Position-Based Visual Servoing

The domain that is used to define commands is a key factor that determines the complexity of the vision system in the feedback segment of the controller. An important consideration in the choice

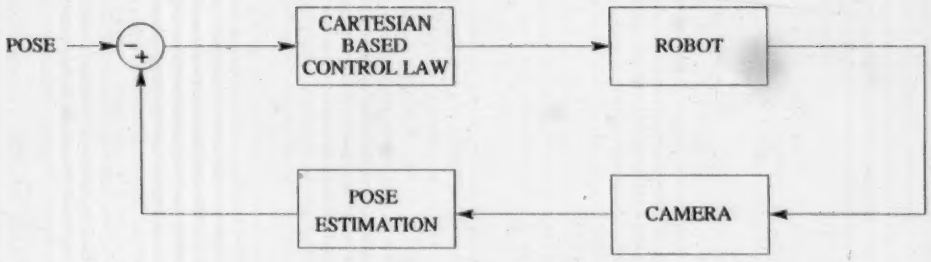


Figure 2.5: Position-based visual servoing

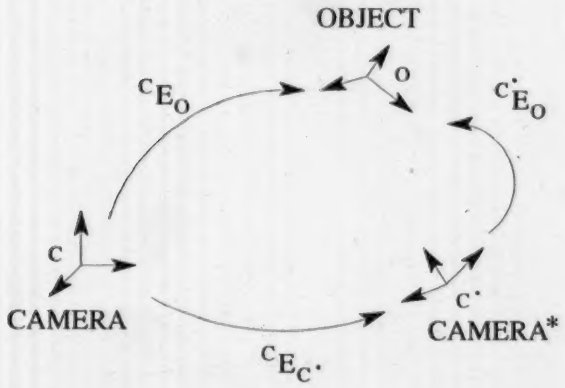


Figure 2.6: Pose estimation.

of the domain is the *a priori* knowledge about the task and about the environment. In practice, relevant knowledge for visual servoing includes the geometry of bodies in the environment and the calibration of the hand-eye system. For applications such as manipulators operating on assembly lines, this knowledge is often available. In industrial environments, the relative orientation and translation between the coordinate frame of the camera and coordinate frame of a body can be estimated and compared to a *position-based* command as illustrated by Figure 2.5. Formally, let the relative orientation and translation between the coordinate frame of a camera C and the coordinate frame of a body O be represented by the homogeneous transformation ${}^{C}E_{O} \in SE(3)$ (Figure 2.6). Given a desired transformation ${}^{C^*}E_{O}$, the task of the control law is to regulate ${}^{C^*}E_{C} = I$, where I denote the identity matrix.

2.5.1 Control Law

Contrary to IBVS, PBVS allows to find "optimal" trajectory in $SE(3)$. This should not come as a surprise since the domain of the PBVS control laws is $SE(3)$. Thus, whereas IBVS control the trajectory of targets in the image space, PBVS control the trajectory of the camera in $SE(3)$. As expected, however, one drawback of PBVS is that it provides little control over the trajectory of image targets. Thus, whereas IBVS does not control the trajectory of the camera in $SE(3)$, PBVS does not control the trajectory of targets in the image space.

Although research in PBVS is more scarce than research in IBVS, several algorithms have been developed to address the challenges of PBVS

Look-then-Move

Perhaps the most intuitive PBVS strategy is known as "look-then-move". Given that ${}^{C}E_{O}$ is estimated from an image, the control law commands the robot with the transformation ${}^{C^*}E_{C}$. If the calibration of the system is accurate, then only one control iteration is required [101]. Otherwise, a new ${}^{C}E_{O}$ is estimated and a new command is used. This formulation is known as look-then-move

because the following estimation of ${}^{C}E_{O}$ is processed after the robot has completed its motion segment. One advantage of this delay is that it provides the vision system a reasonable amount of time to compute ${}^{C}E_{O}$. Depending on the context, an accurate estimation of ${}^{C}E_{O}$ can require several seconds to compute. For example, to pick a body with a gripper requires estimating the pose of the body but also to evaluate different grasps.

Despite its relative simplicity, look-then-move has several advantages. First, because the range of the control law spans $SE(3)$, paths and trajectories can be planned to meet specific needs such as avoiding collision between the manipulator and obstacles. Examples are straight lines in $SE(3)$ with bell-shaped velocity curves. For more complex trajectories, such as quintic splines [125], it is possible to sample $\mathbf{e}_P$ asynchronously and to close the control loop at a higher frequency by splicing the current trajectory with a new trajectory on-line.

First Order Approximation

First order approximation control laws are similar to look-then-move. The main difference between the two methods is that the range of look-then-move typically spans $SE(3)$ whereas the range of first order control laws span the space of twists (generally $\mathbb{R}^6$). For this reason, first order control laws require control loops that operate at higher frequencies.

If ${}^{C}E_{O}$ can be estimated at a sufficiently high rate, then it is possible to formulate the control law as a first order approximation. As with IBVS, this control law is expressed by

$$\mathbf{e}_P = -\lambda L_P^{-1}\mathbf{v} \tag{2.12}$$

where $\mathbf{v}$ is the camera velocity and where $\mathbf{e}_P$

$$\mathbf{e}_P = \begin{bmatrix} {}^{C}\mathbf{t}_O - {}^{C^*}\mathbf{t}_O \\ \theta\mathbf{u} \end{bmatrix} \tag{2.13}$$

represents the translational error $({}^{C}\mathbf{t}_O - {}^{C^*}\mathbf{t}_O)$ and the rotational error (axis $\mathbf{u}$ and angle θ). As a general rule in operation space control, the control of the translation and the rotation are decoupled [24]. In PBVS, this is represented by the 6×6 position-based interaction matrix defined by

$$L_P = \begin{bmatrix} -I_3 & [{}^{C}\mathbf{t}_O]_\times \\ 0 & L_{\theta\mathbf{u}} \end{bmatrix}. \tag{2.14}$$

where $[\,]_\times$ denotes a skew-symmetric matrix. In Equation 2.14, the choice of the rotation control law $L_{\theta\mathbf{u}}$ is defined by [134]

$$L_{\theta\mathbf{u}} = I_3 - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2(\theta)}\right)[\mathbf{u}]_\times . \tag{2.15}$$

It is also possible to formulate the rotational control law with other representations. From the Rodrigues formula, it is known that the rotation matrix $R \in SO(3)$ associated with $\theta\mathbf{u}$ is

$$R = \cos(\theta)I + \sin(\theta)[\mathbf{u}]_\times + (1 - \cos(\theta))\mathbf{u}\mathbf{u}^T. \tag{2.16}$$

By differentiating Equation 2.16, Martinet and Gallice [138] have expressed $L_{\theta\mathbf{u}}$ as

$$L_{\theta\mathbf{u}} = -\frac{1}{2}(\mathrm{trace}(R)I_3 - R^T). \tag{2.17}$$

Another strategy is to track a position-based trajectory with the translational error

$$\mathbf{t}(t) = {}^{C}\mathbf{t}_O(t) - {}^{C^*}\mathbf{t}_O \tag{2.18}$$

and on the rotational error

$$\mathbf{u}(t)\sin(\theta(t)) = \frac{1}{2}(\mathbf{r}_1(t) \times \mathbf{r}_1^* + \mathbf{r}_2(t) \times \mathbf{r}_2^* + \mathbf{r}_3(t) \times \mathbf{r}_3^*) \tag{2.19}$$

where $\mathbf{r}_i$ and $\mathbf{r}_i^*$ are the i^{th} columns of the rotation matrices ${}^{C}R_O \in SO(3)$ and ${}^{C^*}R_O \in SO(3)$ respectively [124]. It is also possible to control the rotation through a different parametrization such as roll, pitch and yaw [186].

In general, PBVS methods do not guarantee that the object will remain within the field of view. This is a serious drawback when tracking ${}^{C}E_{O}$ because such methods typically rely on tracking features on the object's surface. One solution to overcome this challenge involve decoupled control of the depth of the object [180] to "zoom out" when the body is near the borders of the field of view.

Stability

From the development of Section 2.4.2, global asymptotic stability for PBVS is obtained if the pose estimation is accurate. This is readily demonstrated by showing that $L_P L_P^{-1} = I_6$ is positive definite. Given that pose estimation incur errors, a more reasonable formulation is to assume that L_P^{-1} is approximated by $\widehat{L_P^{-1}}$. Thus, $L_P \widehat{L_P^{-1}}$ must be positive definite.

Although PBVS has the potential for global asymptotic stability, this is rarely the case given the errors incurred from the pose estimation. Furthermore, not only does the pose uncertainty represent a bias in the formulation of the error $\mathbf{e}_P$, but the pose uncertainty is also present in the formulation of L_P, thus affecting the robustness of PBVS.

2.5.2 Pose Estimation

Whereas extracting features for IBVS is considered an advantage because of the relative ease and robustness, the main challenge in PBVS is to estimate the pose ${}^C E_O$. Estimating ${}^C E_O$ requires knowledge about the geometry of the body and a calibrated camera. The geometry is required to establish a correspondence between image features and 3D landmarks but it is also required because estimating ${}^C E_O$ implies that the body has a known inertial coordinate frame. As with 3D reconstruction methods, the accuracy of ${}^C E_O$ is severely limited by the accuracy of the calibration and the resolution of the cameras.

The main difficulty of PBVS is the estimation of ${}^C E_O$. Sometimes, this problem is reduced to tracking ${}^C E_O$ given that an initial estimate is provided. Several methods have been developed to estimate ${}^C E_O$. Methods are categorized along several criteria such as camera calibration, available geometry and the number of views and assumptions about the object. By itself, pose estimation is a vast area within computer vision and an in depth review of this area is beyond the scope of this thesis. Nevertheless, some of the main methods are reviewed in the next sections for inclusiveness.

Pose Estimation: Single Camera

In general, the estimation of ${}^C E_O$ by using a single camera requires knowledge about the geometry of the observed body and a calibrated camera. A popular algorithm for this is presented in [55]. It is an analytic method that requires four points and proceeds in two steps. The first step estimates ${}^C E_O$ by solving the system of equations of a scaled orthographic projection. The second step is an iterative algorithm in which the estimate is used to re-project the point back on the image plane. Then, the reprojected points are used to compute a new estimate with the same method used in the first step.

Another two step algorithm is presented in [186]. The first step projects points of the object on the image plane according to an initial estimate of the pose. Then, an extended Kalman filter is used to improve the estimate of ${}^C E_O$ by observing the actual projections. In addition to requiring the geometry of the target, the nature of the Kalman filter suggests an adequate initial guess.

Lowe uses nonlinear least squares to fit a parametrized 3D model to image features [122]. First, a model is constructed from the object's occluding boundaries. Then a transformation tree is defined with the root representing the camera's coordinate frame. Each node of the tree represents a Cartesian transformation that is applied to its children and the points of the model are contained in the leafs. The parameters of ${}^C E_O$ are estimated by matching the model to the edges of an image and by minimizing the error between the edges of the image and the edges of the model.

A variation of PBVS is presented in Alhaj, [3]. The author uses optical flow to reconstruct 3D properties of a surface. By assuming a planar surface, the motion of the camera is used to determine the normal of the plane [1]. The task is to regulate the current plane normal to the desired values. The algorithm assumes brightness constancy and small displacements.

In [172], a different approach is used to estimate the pose. A wire-frame representation of the scene is used in a virtual world. The pose of the camera is estimated in the virtual world and is used to track the pose of the camera in reality. The non-linear estimation is solved by the Gauss-Newton algorithm.

In [4], stereo optical flow is used to estimate the 3D coordinates of a moving object. The control strategy is used to predict the coordinates of the object in order to account for computing latency. Experiments consisted of picking up a circling miniature train.

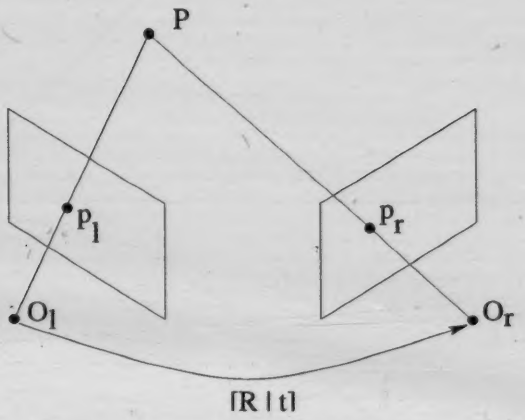


Figure 2.7: Two views geometry.

Pose Estimation: Multiple Cameras

It is possible to recover three dimensional information of a scene if two or more views are used. The accuracy of such reconstructions depends on the camera model used to represent image formation as well as the accuracy of the cameras calibration. This section only discusses pose estimation methods based on the *pinhole* camera model and presents a brief summary of the theory behind three dimensional reconstruction.

Let the coordinate frames l and r of two cameras be related by a rotation matrix $R \in SO(3)$ and a translation vector $\mathbf{t} \in \mathbb{R}^3$ as illustrated by Figure 2.7.

A 3D point $\mathbf{P}$ has the coordinates ${}^l\mathbf{P} = \begin{bmatrix} {}^lX & {}^lY & {}^lZ \end{bmatrix}^T$ in the coordinate frame l and the coordinates ${}^r\mathbf{P} = \begin{bmatrix} {}^rX & {}^rY & {}^rZ \end{bmatrix}^T$ in the coordinate frame r. It follows that

$$ {}^l\mathbf{P} = R({}^r\mathbf{P} - \mathbf{t}). \tag{2.20} $$

The projective coordinates of each point in their respective coordinate frame are given by

$$ {}^l\mathbf{p} = \begin{bmatrix} {}^lx \\ {}^ly \\ {}^lz \end{bmatrix} = {}^lK_l{}^l\mathbf{P}; \qquad {}^r\mathbf{p} = \begin{bmatrix} {}^rx \\ {}^ry \\ {}^rz \end{bmatrix} = {}^rK_r{}^r\mathbf{P} \tag{2.21} $$

where K contains the *intrinsic parameters* of a camera

$$ K = \begin{bmatrix} -f/s_x & s & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.22} $$

The intrinsic parameters are defined by the focal length f of the lens, the horizontal and vertical scaling s_x, s_y of the CCD array, the coordinates of the optical axis (o_x, o_y) and camera skew s. Many calibration methods have been researched to estimate these parameters [89] and most of them require calibration patterns or structured environments.

The actual coordinates in the images are determined by

$$ {}^l\mathbf{p}_i = \begin{bmatrix} {}^lx_i \\ {}^ly_i \end{bmatrix} = \begin{bmatrix} \frac{{}^lx}{{}^lz} \\ \frac{{}^ly}{{}^lz} \end{bmatrix} \qquad {}^r\mathbf{p}_i = \begin{bmatrix} {}^rx_i \\ {}^ry_i \end{bmatrix} = \begin{bmatrix} \frac{{}^rx}{{}^rz} \\ \frac{{}^ry}{{}^rz} \end{bmatrix}. \tag{2.23} $$

If the intrinsic parameters ${}^lK_l$, ${}^rK_r$, the rotation R and the translation $\mathbf{t}$ are known, then the coordinates of ${}^l\mathbf{P}$ (or ${}^r\mathbf{P}$ if the coordinate frame r is chosen as the reference) can be determined by triangulation. If the intrinsic parameters of both cameras are known, but R and $\mathbf{t}$ are not, the

Calibration	3D reconstruction
Intrinsic and extrinsic parameters	Absolute coordinates
Intrinsic parameters	Up to an unknown scale factor
No parameters	Up to an unknown projective transformation

Table 2.1: Geometric reconstruction versus camera calibration.

coordinates of ${}^{l}\mathbf{P}$ can be recovered up to an unknown scale. Using epipolar geometry [89], the *essential matrix* E relates the two 3D coordinates by

$$ {}^{r}\mathbf{P}^{T} E {}^{l}\mathbf{P} = 0. \tag{2.24} $$

in which essential matrix E is defined by

$$ E = RT_{\times}, $$

and where $T_{\times}$ is the skew symmetric matrix of $\mathbf{t}$. Substituting equation (2.21) in equation (2.24) results in

$$ \mathbf{p}_{r}^{T} E \mathbf{p}_{l} = 0. \tag{2.25} $$

The solution to find E requires a system of at least 8 equations. Then, from E, it is possible to recover the normalized translation $\mathbf{t} = \mathbf{t}/\|\mathbf{t}\|$ and the rotation R [181]. Consequently, the 3D reconstruction of a scene is only up to a unknown scale factor. This scale factor can be determined if the distance between two points is provided.

If the cameras are not calibrated, the 3D coordinates can only be recovered up to an unknown projective transformation. Using equation (2.23) in (2.25) we get

$$ {}^{r}\mathbf{p}^{T} F {}^{l}\mathbf{p} = 0 \tag{2.26} $$

where the *fundamental matrix* F is defined by

$$ F = {}^{r}K_{r}{}^{T} E {}^{l}K_{l}. $$

As with the essential matrix, at least 8 corresponding points are required to estimate F [88]. Using F, a projective reconstruction is determined along the standard projective basis ($[0\ 0\ 0\ 1]^{\top}$, $[0\ 0\ 1\ 0]^{\top}$, $[0\ 1\ 0\ 0]^{\top}$, $[1\ 0\ 0\ 0]^{\top}$, $[1\ 1\ 1\ 1]^{\top}$) derived from the combinations of five image points. The unknown projective transformation can be determined given the Cartesian coordinates of the five points associated used to determine the basis.

Table 2.1 summarizes the structures that can be determined given the level of camera calibration.

Because robots move in a Cartesian space, the cameras must be fully calibrated to implement PBVS based on 3D reconstruction. Within this context, the body's inertial coordinate frame must be determined from the estimated 3D points and the relative pose between this coordinate frame and the coordinate frame of a camera is used to regulate the manipulator.

Pose Estimation: Vision-Based Localization

The problem of localization and the related problem of simultaneous localization and mapping (SLAM) is often encountered in mobile robotics. Despite the focus on mobile robots, the problems addressed in these areas are similar to the more general problem of pose estimation. In particular, the problem of localization of a mobile robot is defined as determining its coordinates on a floor and its orientation θ with respect to an inertial coordinate frame. Typically, these parameters are computed by estimating the 3D coordinates of visual landmarks and by comparing the coordinates to those stored in a map. From the two sets of 3D coordinates, the translation and heading of the robot can be estimated by various methods [21, 22]. The map is either provided beforehand or constructed while exploring the environment.

Because odometry measurements are unreliable, most successful methods use filters to estimate the position s of a robot from an observation o. In particular, Bayes filter, defined by

$$ P(s|o) = \frac{P(o|s)P(s)}{P(o)}, \tag{2.27} $$

represents the probability of being in a position s, given an observation o. In a dynamic system, a sequence of motions a_t are performed and a sequence of observations o_t are measured. Markov localization methods seek to evaluate the posterior probability at each time interval t

$$P(s_t|o_1, \ldots, o_t, a_1, \ldots, a_{t-1}). \tag{2.28}$$

by using the Markov assumption,

$$P(s_t|s_1, \ldots, s_{t-1}, a_1, \ldots, a_{t-1}) = P(s_t|s_{t-1}, a_{t-1}). \tag{2.29}$$

The solution to equation (2.28) is evaluated in three steps

1. Prediction
$$\hat{P}(s_t) = \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) P(s_{t-1}) \tag{2.30}$$

2. Update
$$P(s_t) = \alpha P(o_t|s_t) \hat{P}(s_t) \tag{2.31}$$

3. Estimate
$$s_t^* = MAP(P(s_t)) \tag{2.32}$$

Depending on the model used to represent the probability distribution and the domain, several variants of the above *Markov localization* framework were developed. Markov localization is mostly used with discrete spaces [31, 73, 110, 169, 179]. Kalman filters were used in cases where the distribution is Gaussian and linearized dynamics and observation models are used [113, 166, 171]. Similarly, several methods remove the linear constraints by using extended Kalman filters [51] A powerful approximation method to handle multi-modal distribution and partially observable states is importance sampling [60, 121]. Localization methods based on samples, also known as Monte Carlo localization, were successfully used in [54, 72]. Several other methods have combined sampling methods with Rao-Blackwellised estimators instead of the maximum *a posteriori* (MAP) estimator [61].

Ego Motion

Similarly to pose estimation, ego motion estimation refers to estimating the camera displacement between two views. The nuance stems from the assumption that ego motion typically assumes small variations between frames. Therefore, several methods rely on dense flow field, which is not always available or reliable.

A simple and efficient ego motion estimation method is presented in [87]. Kalman filters are used to track corners and recovers the ego motion and the 3D coordinates of the corners. Irani *et al.* derive the motion equations for points on a plane to compute region alignment of planar surfaces and recover 3D camera motion [102]. The parameters of the motion found by employing a search algorithm bounded by geometric constraints[70]. Heeger and Jepson use a subspace method to partition the flow field with three sets of equations representing translation, rotation and depth [94].

2.5.3 Hand-Eye Calibration

Another area of research related to visual servoing is hand-eye calibration. Hand-eye calibration is related to camera calibration and multiple view geometry discussed in previous section. It seeks to estimate the extrinsic parameters X of the camera with respect to the coordinate frame of the end-effector. This calibration is essential for the control law to make sense of any measurements captured by a sensor fixed on an end-effector.

Let ${}^{B}E_{C}$ denote the homogeneous transformation between the coordinate frame of the camera and coordinate frame of the base of the robot. Also, let ${}^{B}E_{T6}$ be the transformation from the robot base to the end-effector. Given that ${}^{B}E_{C}$ is determined from the extrinsic parameters of the camera and ${}^{B}E_{T6}$ from the forward kinematics, X is the solution to the *hand-eye equation* [37]

$${}^{B}E_{C} X = X {}^{B}E_{T6}. \tag{2.33}$$

Evaluating the solution to equation (2.33) requires two views.

Research on hand-eye calibration is concerned by computing a reliable solution to equation (2.33). Proposed methods include linear solutions [188], closed-form [59, 37], nonlinear optimization [59]. Other methods have employed structure from motion [7] or recursive least-squares that can be applied on line [8].

It is important to note that visual servoing systems must account for this hand-eye calibration. For example, in PBVS, the relative pose of the body is expressed in the coordinate frame of the camera as opposed to the coordinate frame of the end-effector. Accordingly, in IBVS, the velocity corresponds to the body velocity of the camera, not the velocity of the end-effector. Thus, the coordinate frame of the camera must be configured as the tool control point of the robot in order for the output of any control law to be applied to the appropriate coordinate frame.

2.5.4 Learning and Sensory Control

Learning methods have also been used in sensory control methods. Several of them are used within the context of embodiment [25] for high level interaction with humanoids.

A hand-eye system for dropping objects on the wagons of a miniature train is presented in [30]. The hand-eye coordination is learned by moving a light bulb at some specific points on a mesh grid and recording the position of the bulb on the table and the state of the robot. The motion of the robot in the workspace is then determined by interpolating over the look up table.

A 2 DOF stereo algorithm that learns to track a target is introduced in [150]. The method trains a neural network with self-organizing fovea. The fovea is characterized with a higher density of units in area that are within the reach of the first corrective motion (saccade).

In [18], the authors experiments with various approximations to learn non-linear controllers used for robotics applications. Among the approximators are: multilayer perceptrons, radial basis function network and fuzzy controllers. Iterative learning control [9] is used to approximate the linearity around a point of a nonlinear function. Iterative control learning is also used for a position/force controller in [16].

Learning has also been used in grasping and manipulation. In [163], the robot learns a grasping approach for objects. An object is represented by its three semi-major axis. The axes are extracted from range images by fitting a super-quadratic surface to the data. Then the interval estimation algorithm is used to keep track of the success rate of each action and a confidence interval is assigned to each rate. Finally, the IE algorithm is extended to a continuous space by using a decision tree. RBF neural network was used in another grasping research [156]. The network performs an appearance based object recognition and evaluates a grasp.

Baroglio *et al.* used learning to control every aspect of an industrial robot. Their effort involves approximating non-linear continuous functions with neural networks. Among the types tested are multilayer perceptrons, radial basis functions and fuzzy controllers.

Learning has also been applied to high dimension spaces. In [74], a Utah/MIT hand with 16 DOF is used for dexterous manipulation. To reduce the dimensionality, the authors use *virtual fingers* in order to represent a manipulation with a minimal number of DOF (even though all DOF are actually used to perform the manipulation). This representation is used to learn manipulation primitives on objects. Then, the primitives are used on different objects and the appropriate parameters are found with a nearest neighbor algorithm. The learning is based on the evolution strategy.

Target reaching and pointing using learning is proposed in [136, 58]. In [136], the humanoid learned to locate its arm in the images and track a visual target. Learning was used to create a map between the orientation of the gaze and the arm state in order to move the end effector at the center of the visual field. In [58], Q-learning was used to reach and grasp a spherical object randomly placed in the workspace. To cope with the dimensionality of state-action spaces, the task is broken into a sequence of sub-tasks, namely centering and approaching, each with their own discrete action and state spaces.

A method for a mobile robot to learn how to shoot a ball into a goal is presented in [11]. A state is defined by discretizing the relative position and size of the ball with respect to the robot and the relative position, size and orientation of the goal. The action space is defined by 9 commands controlling the wheels. Q-learning is used to estimate the action value function.

A handful of learning methods have been specifically applied to visual servoing. A learning approach has also been used to estimate the image Jacobian [104]. In [76, 78, 77, 178], Q-learning is used for visual servoing. The authors point out several merits of using learning, namely that calibrations of sensors and actuators are not required. Furthermore, they demonstrate the flexibility of their method by using on a mobile robot and an underwater vehicle.

Chapter 3

Visuomotor Function

The interaction matrix used in IBVS and PBVS involves a first order approximation. This approximation relates the exponential decrease of the error vector $\mathbf{e}$ to the velocity of the camera. For IBVS, the interaction matrix relates velocities of image features to the camera velocity, even though the minimization is done by using the errors of features instead the errors velocity. Although the IBVS interaction matrix has several advantages its domain limits the planning of tasks and trajectories. A drawback of IBVS is that it is not suitable for feed-forward position commands that are expected by several manipulators, especially industrial ones. The trajectories in feature space or task space can conform to some guidelines or rules, but these trajectories are generated on-line such that a motion cannot be planned ahead of its execution. Furthermore, IBVS also requires that a minimum number of features remain visible throughout the entire motion. This reduces the range of tasks that a robot can perform as the targets must remain visible. That is each target must stay within the field of view of the camera and must not become occluded during the task. It is important to note that these restrictions are not caused by the range of IBVS interaction matrices but by their domain. From the literature, the range of an IBVS interaction matrix is the velocity of image-based errors.The domain, however, consists of the space of twists ($\mathbb{R}^6$) and planning motion in $SE(3)$ from this space is counter intuitive.

Instead of deriving an IBVS interaction matrix whose domain is the space of twists, this chapter introduces an interaction matrix called the visuomotor function whose domain is the special Euclidean space $SE(3)$ and the co-domain are variations of image coordinates

$$V : SE(3) \rightarrow \mathbb{R}^d.$$

Thus, V aims at combining the advantages of range of IBVS interaction matrices with the domain of PBVS look-then-move.

3.1 Visuomotor Function

Following the visual servoing nomenclature, the visuomotor function formulation can be derived from an *eye-in-hand* or a *eye-to-hand* configuration. In a typical eye-in-hand configuration, the camera is mounted on the end-effector and the control law controls the motion of the camera's coordinate frame. In a eye-to-hand configuration, the camera is mounted on a separate base and the control law controls the motion of the robots tool control point (TCP).

Because the visuomotor function relates variations of image coordinates to variation in $SE(3)$ it is defined for a single camera. But for reasons that will be explained in Chapter 4, two cameras will be necessary. For this reason, the equations presented in this chapter use the L and R superscripts to distinguish between the right and left cameras even though the theory presented in this chapter only concerns a single camera.

The illustration for an eye-in-hand configuration used to derive the visuomotor function is presented in Fig. 3.1. The configuration consists of two cameras, left and right, with their respective coordinate frames L and R. The positions of each camera relative to the coordinate frame of a stereo frame S is defined according to the homogeneous transformations ${}^{L}E_{S}$ and ${}^{R}E_{S}$. The frame S is attached to the end-effector of the robot. Since ${}^{L}E_{S}$ and ${}^{R}E_{S}$ are arbitrary, the coordinate frame of the end-effector coincides with S. The coordinate frame B represents the base of the robot and it relates to the frame S by the robot's kinematics denoted by ${}^{S}E_{B}$. Since ${}^{S}E_{B}$ is obtained from the kinematics the following derivations are expressed in the frame S.

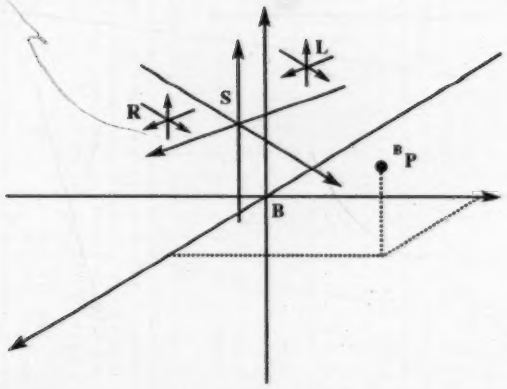


Figure 3.1: Geometric model: Camera frames L and R are mounted on a stereo rig S.

The objective of the visuomotor function is to relate *arbitrary* variations of image coordinates to a *relative transformation* of the end-effector's coordinate frame. For conciseness, equations will be derived only for the left camera. Derivations for the right camera are identical in every way.

The first step is to derive the projective coordinates in the left camera, denoted ${}^{L}\mathbf{p}$, of a static 3D point with homogeneous coordinates ${}^{B}\mathbf{P} = \begin{bmatrix} {}^{B}X & {}^{B}Y & {}^{B}Z & 1 \end{bmatrix}^{T}$. The second step is to derive an expression relating the variation of ${}^{L}\mathbf{p}$ as a function of the relative rigid displacement of the frame S from ${}^{S}E_{B}$ to ${}^{S}E'_{B}$ according to the homogeneous transformation

$$ {}^{S}E'_{B}\,{}^{B}E_{S} = {}^{S}E'_{S} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where $R \in SO(3)$ is the rotation and $\mathbf{t} \in \mathbb{R}^3$ is the translation.

First, the coordinates of ${}^{B}\mathbf{P}$ are transformed in the coordinate frame S by ${}^{S}E_{B}$ and in the coordinate frame of the (left) camera L by ${}^{L}E_{S}$. Second, under the conventional pinhole camera model, ${}^{L}\mathbf{P}$ is projected on the image plane according to the projection matrix K (Equation 2.22) such that

$$ {}^{L}\mathbf{p} = \begin{bmatrix} {}^{L}x \\ {}^{L}y \\ {}^{L}z \end{bmatrix} = K\,{}^{L}E_{S}\,{}^{S}E_{B}\,{}^{B}\mathbf{P}. \tag{3.1} $$

From Equation 3.1, let matrix M be the combination of intrinsic and extrinsic parameters

$$ M = {}^{L}K_{L}\,{}^{L}E_{S} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} \tag{3.2} $$

Furthermore, given the initial position and orientation of the end-effector ${}^{S}E_{B}$ and noting that ${}^{S}\mathbf{P} = {}^{S}E_{B}\,{}^{B}\mathbf{P}$ we get

$$ {}^{L}\mathbf{p} = M\,{}^{S}\mathbf{P} \tag{3.3} $$

The second step is to express the image coordinates of ${}^{B}\mathbf{P}$ after the displacement of the stereo frame S according to the transformation ${}^{S}E'_{B}$. Substituting ${}^{S}E_{B}$ for ${}^{S}E'_{B}$ in Equations 3.1 results in

$$ {}^{L}\mathbf{p}' = \begin{bmatrix} {}^{L}x' \\ {}^{L}y' \\ {}^{L}z' \end{bmatrix} = M\,{}^{S}E'_{B}\,{}^{B}\mathbf{P}. \tag{3.4} $$

Again, noting that ${}^{S}E'_{B}\,{}^{B}\mathbf{P} = {}^{S}E'_{S}\,{}^{S}E_{B}\,{}^{B}\mathbf{P} = {}^{S}E'_{S}\,{}^{S}\mathbf{P}$, Equation 3.4 becomes

$$ {}^{L}\mathbf{p}' = M\,{}^{S}E'_{S}\,{}^{S}\mathbf{P}. \tag{3.5} $$

The visuomotor function is derived from the difference between Equation 3.5 and Equation 3.3. Because the projective space is not closed under subtraction, the difference between ${}^L\mathbf{p}'$ and ${}^L\mathbf{p}$ is determined by transforming the homogeneous points in the real space, perform the subtraction, and transform the result back in the projective space. That is, if the image coordinates are defined by

$${}^L\mathbf{p}_i = \begin{bmatrix} \frac{{}^Lx}{{}^Lz} \\ \frac{{}^Ly}{{}^Lz} \end{bmatrix}; \quad {}^L\mathbf{p}'_i = \begin{bmatrix} \frac{{}^Lx'}{{}^Lz'} \\ \frac{{}^Ly'}{{}^Lz'} \end{bmatrix}; \quad ,$$

then we obtain

$${}^L\mathbf{p}'_i - {}^L\mathbf{p}_i = \begin{bmatrix} \frac{{}^Lx'}{{}^Lz'} - \frac{{}^Lx}{{}^Lz} \\ \frac{{}^Ly'}{{}^Lz'} - \frac{{}^Ly}{{}^Lz} \end{bmatrix} = \begin{bmatrix} \frac{{}^Lz{}^Lx' - {}^Lz'{}^Lx}{{}^Lz'{}^Lz} \\ \frac{{}^Lz{}^Ly' - {}^Lz'{}^Ly}{{}^Lz'{}^Lz} \end{bmatrix}.$$

Thus, from this operation, we obtain the variation $\Delta^L\mathbf{p} = {}^L\mathbf{p}' - {}^L\mathbf{p}$

$$\Delta^L\mathbf{p} = \begin{bmatrix} \Delta^Lx \\ \Delta^Ly \\ \Delta^Lz \end{bmatrix} \equiv \begin{bmatrix} {}^Lz{}^Lx' - {}^Lx{}^Lz' \\ {}^Lz{}^Ly' - {}^Ly{}^Lz' \\ {}^Lz{}^Lz' \end{bmatrix}. \tag{3.6}$$

Expanding Equation 3.6 results in

$$\Delta^L\mathbf{p} = \begin{bmatrix} \mathbf{m}_3{}^S\mathbf{P}\mathbf{m}_1 - \mathbf{m}_1{}^S\mathbf{P}\mathbf{m}_3 \\ \mathbf{m}_3{}^S\mathbf{P}\mathbf{m}_2 - \mathbf{m}_2{}^S\mathbf{P}\mathbf{m}_3 \\ \mathbf{m}_3{}^S\mathbf{P}\mathbf{m}_3 \end{bmatrix} {}^SE'_S{}^S\mathbf{P} = N({}^L\mathbf{p}){}^SE'_S{}^S\mathbf{P} \tag{3.7}$$

where N is called the *visuomotor camera* and is represented by

$$N({}^L\mathbf{p}) = \begin{bmatrix} {}^Lz\mathbf{m}_1 - {}^Lx\mathbf{m}_3 \\ {}^Lz\mathbf{m}_2 - {}^Ly\mathbf{m}_3 \\ {}^Lz\mathbf{m}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{n}_1 \\ \mathbf{n}_2 \\ \mathbf{n}_3 \end{bmatrix}. \tag{3.8}$$

Equation 3.7 is called the *visuomotor function* as it relates the relative displacement of the end-effector ${}^SE'_S$ to variations of image coordinates $\Delta^L\mathbf{p}$. It is important to note that ${}^S\mathbf{P}$ is defined in the coordinate frame of the end-effector (S) such that any reference to the inertial frame B are avoided. Equation 3.7 has the same form than the usual projection equation (Equations 3.3). The main differences are that Equation 3.7 defines a projective variation instead of projective coordinates and the parameters of the visuomotor camera depend on the projective coordinates ${}^L\mathbf{p}$. Thus, the visuomotor camera is an active camera that *projects variations of coordinates* resulting from the displacement ${}^SE'_S$ of a point ${}^S\mathbf{P}$. Comparatively, conventional camera model projects the coordinates of a point, not their variations.

One observation about Equation 3.7 is that it projects the variation of ${}^S\mathbf{P}$ given a motion of the end-effector. As expected, this projection depends on the parameters of the actual camera but also on the projective coordinates of ${}^L\mathbf{p}$. Also, Equation 3.7 is defined up to a scale of both sides. The right hand side will be analyzed in greater details in Section 3.3, but it is easy to demonstrated that given a vector $\Delta^L\mathbf{p}$ and a transformation ${}^SE'_S$ then, for a constant κ, any $(\kappa N({}^L\mathbf{p}), {}^S\mathbf{P}/\kappa)$ is a possible solution to Equation 3.7. Interestingly, $\Delta^L\mathbf{p}$ is a projective vector and, thus, it is also defined up to a scale $\Delta^L\mathbf{p} = \kappa\Delta^L\mathbf{p}$.

A final observation about Equation 3.7 is that motion constraints imposed by a task are reflected by the degrees of freedom (DOF) in the rotation R and the translation $\mathbf{t}$. These constraints lead to the cancellation of elements in $N({}^L\mathbf{p})$ and ${}^S\mathbf{P}$. The following sections will present variants of the visuomotor function for a pan-tilt unit, a mobile robot, 3D translations and a 6DOF motion.

3.1.1 6DOF Motion

To use the visuomotor system, the visuomotor camera $N({}^L\mathbf{p})$ must be calibrated and the coordinates of ${}^S\mathbf{P}$ must be known. In the coming sections, these parameters will be obtained by approximating the visuomotor system from pairs of observations $({}^SE'_S, \Delta^L\mathbf{p})$. For now, it is possible to expand

the products in Equation 3.7 to obtain a linear formulation. In Equation 3.7, this is done by coupling the parameters of $N({}^{L}\mathbf{p})$ with the coordinates of ${}^{S}\mathbf{P}$. As a result, $\Delta^{L}\mathbf{p}$ will be expressed by a linear combination between a matrix V and the elements of ${}^{S}E'_{S}$.

In this thesis, the coupling of the parameters of $N({}^{L}\mathbf{p})$ with those of ${}^{S}E'_{S}$ are called the *visuomotor parameters* since they relate directly the variation of the motor system to variations of the visual system.

In general, the end-effector is free to move along and around any axis. For this general case, the visuomotor function for unconstrained motion is derived directly from Equation 3.7. Equation 3.7 is composed of one bilinear term and a total of 15 parameters (12 parameters in $N({}^{L}\mathbf{p})$ and 3 in ${}^{S}\mathbf{P}$). Writing Equation 3.7 in terms of the inhomogeneous coordinates the image-based error is defined as follow

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \frac{{}^{L}x'}{{}^{L}z'} - \frac{{}^{L}x}{{}^{L}z} \\ \frac{{}^{L}y'}{{}^{L}z'} - \frac{{}^{L}y}{{}^{L}z} \end{bmatrix} = \begin{bmatrix} \frac{\Delta^{L}x}{\Delta^{L}z} \\ \frac{\Delta^{L}y}{\Delta^{L}z} \end{bmatrix}$$

After a manipulation of Equation 3.7 we obtain the homogeneous equations

$$0 = \mathbf{n}_1 {}^{S}E'_{S} {}^{S}\mathbf{P} - e_x \mathbf{n}_3 {}^{S}E'_{S} {}^{S}\mathbf{P} \tag{3.9}$$

$$0 = \mathbf{n}_2 {}^{S}E'_{S} {}^{S}\mathbf{P} - e_y \mathbf{n}_3 {}^{S}E'_{S} {}^{S}\mathbf{P}. \tag{3.10}$$

The linear formulation of Equations 3.9 and 3.10 generates 26 parameters per equation (13 parameters for each bilinear term). Noting that the second bilinear term in both equations are equal up to a scale, the 13 parameters associated with them can be shared by both equations by factoring out e_x and e_y. Hence, this gives the choice of solving two systems of 26 unknowns or one system of 39 unknowns.

The homogeneity of Equations 3.9 and 3.10 can be removed given that

$$\mathbf{n}_3 = \begin{bmatrix} n_{31} & n_{32} & n_{33} & n_{34} \end{bmatrix}$$

and

$$e_x \mathbf{n}_3 {}^{S}E'_{S} {}^{S}\mathbf{P} = e_x \begin{bmatrix} n_{31} & n_{32} & n_{33} \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \end{bmatrix} {}^{S}\mathbf{P} + e_x n_{34} \tag{3.11}$$

$$e_y \mathbf{n}_3 {}^{S}E'_{S} {}^{S}\mathbf{P} = e_y \begin{bmatrix} n_{31} & n_{32} & n_{33} \end{bmatrix} \begin{bmatrix} R & \mathbf{t} \end{bmatrix} {}^{S}\mathbf{P} + e_y n_{34}. \tag{3.12}$$

If $n_{34} \neq 0$, then substituting Equations 3.11 and 3.12 in 3.9 and 3.10 and solving for e_x and e_y results in inhomogeneous equations

$$e_x = \mathbf{o}_1 {}^{S}E'_{S} {}^{S}\mathbf{P} - e_x \mathbf{o}_3 {}^{S}E'_{S} {}^{S}\mathbf{P} \tag{3.13}$$

$$e_y = \mathbf{o}_2 {}^{S}E'_{S} {}^{S}\mathbf{P} - e_y \mathbf{o}_3 {}^{S}E'_{S} {}^{S}\mathbf{P} \tag{3.14}$$

where

$$\mathbf{o}_1 = \frac{\mathbf{n}_1}{n_{34}}; \quad \mathbf{o}_2 = \frac{\mathbf{n}_2}{n_{34}}; \quad \mathbf{o}_3 = \frac{\begin{bmatrix} n_{31} & n_{32} & n_{33} & 0 \end{bmatrix}}{n_{34}}. \tag{3.15}$$

Multiplying out the bilinear terms we obtain the following matrix form of the 6 DOF visuomotor function

$$\mathbf{e} = V \begin{bmatrix} {}^{S}\mathbf{E}'_{S} \\ 1 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \theta_1 & \cdots & \theta_{13} \end{bmatrix} - e_x \begin{bmatrix} \theta_{27} & \cdots & \theta_{38} & 0 \end{bmatrix} \\ \begin{bmatrix} \theta_{14} & \cdots & \theta_{26} \end{bmatrix} - e_y \begin{bmatrix} \theta_{27} & \cdots & \theta_{38} & 0 \end{bmatrix} \end{bmatrix} \begin{bmatrix} {}^{S}\mathbf{E}'_{S} \\ 1 \end{bmatrix} \tag{3.16}$$

where ${}^{S}\mathbf{E}'_{S}$ is a column vector composed of the elements of ${}^{S}E'_{S}$

$${}^{S}\mathbf{E}'_{S} = \begin{bmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & \mathbf{r}_3^T & \mathbf{t}^T \end{bmatrix}^T, \tag{3.17}$$

and where the visuomotor parameters are defined by

$$\begin{aligned}
\theta_1 = o_{11}{}^S X; \quad \theta_2 = o_{12}{}^S X; \quad \theta_3 = o_{13}{}^S X; \\
\theta_4 = o_{11}{}^S Y; \quad \theta_5 = o_{12}{}^S Y; \quad \theta_6 = o_{13}{}^S Y; \\
\theta_7 = o_{11}{}^S Z; \quad \theta_8 = o_{12}{}^S Z; \quad \theta_9 = o_{13}{}^S Z; \\
\theta_{10} = o_{11}; \quad \theta_{11} = o_{12}; \quad \theta_{12} = o_{13}; \quad \theta_{13} = o_{14}; \\
\theta_{14} = o_{21}{}^S X; \quad \theta_{15} = o_{22}{}^S X; \quad \theta_{16} = o_{23}{}^S X; \\
\theta_{17} = o_{21}{}^S Y; \quad \theta_{18} = o_{22}{}^S Y; \quad \theta_{19} = o_{23}{}^S Y; \\
\theta_{20} = o_{21}{}^S Z; \quad \theta_{21} = o_{22}{}^S Z; \quad \theta_{22} = o_{23}{}^S Z; \\
\theta_{23} = o_{21}; \quad \theta_{24} = o_{22}; \quad \theta_{25} = o_{23}; \quad \theta_{26} = o_{24}; \\
\theta_{27} = o_{31}{}^S X; \quad \theta_{28} = o_{32}{}^S X; \quad \theta_{29} = o_{33}{}^S X; \\
\theta_{30} = o_{31}{}^S Y; \quad \theta_{31} = o_{32}{}^S Y; \quad \theta_{32} = o_{33}{}^S Y; \\
\theta_{33} = o_{31}{}^S Z; \quad \theta_{34} = o_{32}{}^S Z; \quad \theta_{35} = o_{33}{}^S Z; \\
\theta_{36} = o_{31}; \quad \theta_{37} = o_{32}; \quad \theta_{38} = o_{33};
\end{aligned} \tag{3.18}$$

3.1.2 Pan-Tilt Unit

This sections considers the visuomotor function of stereo cameras mounted on the end-effector of a pan-tilt unit (PTU). A PTU consists of two DOF that orient a coordinate frame. In practice, PTUs also involves a 3D translation of the end-effector because of the design of the kinematic chains. In a typical PTU (Fig. 3.2), the PAN actuator moves the tilt link, which also includes the $TILT$ actuator. Because of the physical presence of an actuator on each link, it is difficult to configure two actuators in order for them to act on the end-effector without involving any translation. This is illustrated by the modified DH [109] diagram of a PTU in Fig. 3.3. The frame S correspond to the end-effector of the PTU on which the left camera L is mounted. The PTU is composed of the coordinate frames of the pan (P) and tilt (T) links. The length of the tilt link generates a translation of the frame S with respect to the base frame and this translation must be accounted by ${}^S E_B$ to represent the true motion of the cameras with respect to the coordinate frame B. In the formulation used to derive the visuomotor function, however, because the transformations ${}^L E_S$ and ${}^R E_S$ are arbitrary, the coordinate frame S can be displaced by a constant transformation. Thus, the length of tilt link can be absorbed by ${}^L E_S$ and ${}^R E_S$ such that the origin of frame S can be moved down and coincide with the origin of frame P.

This property of positioning arbitrarily the coordinate frame S provides three advantages. First, it avoids complex kinematic designs that do pure rotations of a camera frame [83]. Second, the forward kinematics of the PTU does not need to be known and evaluated to determine the position of S and, consequently, of L. Finally, the transformation ${}^S E'_S$ is reduced from 5 DOFs (2 DOFs for the explicit rotation and 3 DOFs for the implicit translation) to 2 DOFs rotational motion.

Most commercial PTUs, as the one depicted in Figure 3.2, adopt a Z-Y Euler angles to parametrize rotations (Figure 3.4). That is, the coordinate frame rotates around the Z axis by an angle θ_z, followed by a rotation around the Y axis by an angle θ_y. As with all rotations parametrized by Euler angles, the order of the rotations matters. In this case, the transformation ${}^S E'_S$ is represented by

$${}^S E'_S(\theta_z, \theta_y) = \begin{bmatrix} R(\theta_z, \theta_y) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & 0 & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.19}$$

Using the pan-tilt model, Equation 3.7 becomes

$$\mathbf{e} = N({}^L\mathbf{p}) \begin{bmatrix} R(\theta_z, \theta_y) & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} {}^S\mathbf{P} \tag{3.20}$$

From the 6 DOF visuomotor function, the unused terms ($\mathbf{t}_z$ and r_{23}) are removed such that we

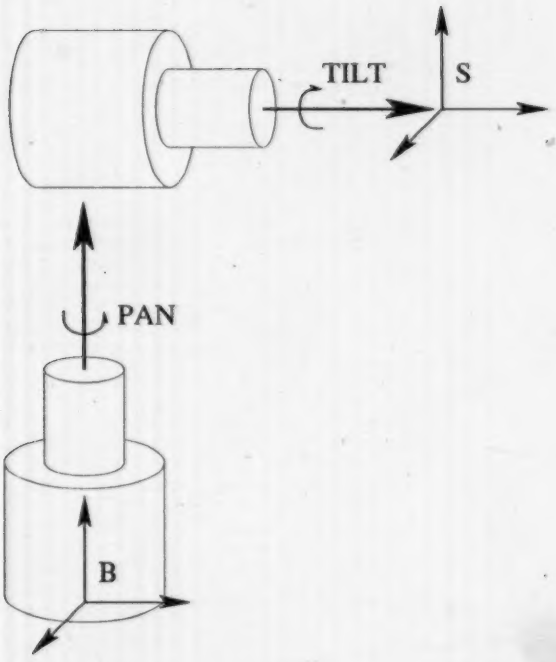


Figure 3.2: Kinematics of a pan-tilt unit: A typical PTU involves a translation of its end-effector.

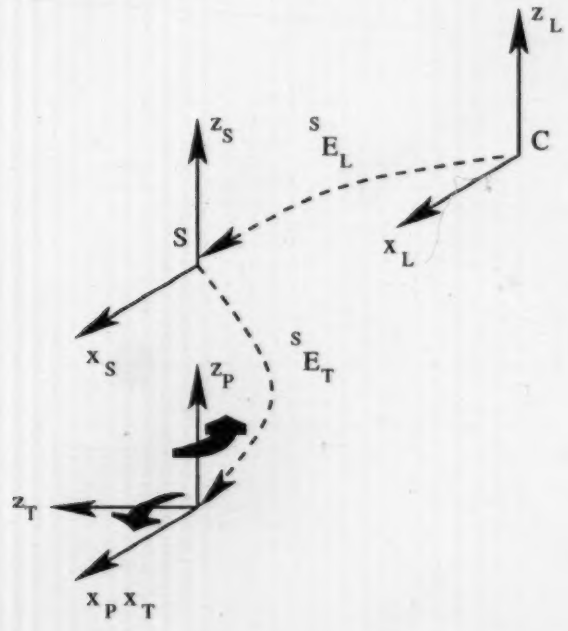


Figure 3.3: Modified Denavit-Hartenberg diagram of a PTU mounted with camera (frame C).

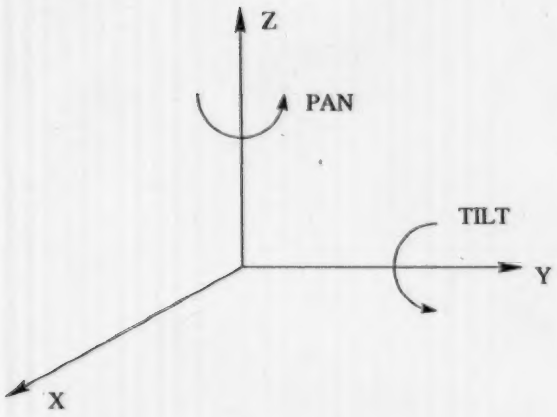


Figure 3.4: Z-Y Euler angles.

obtain

$$e_x = \begin{bmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & r_{13} & r_{33} & 1 & -e_x\mathbf{r}_1^T & -e_x\mathbf{r}_2^T & -e_x r_{13} & -e_x r_{33} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_7 \\ \theta_9 \\ \theta_{13} \\ \theta_{27} \\ \vdots \\ \theta_{33} \\ \theta_{35} \end{bmatrix} \tag{3.21}$$

$$e_y = \begin{bmatrix} \mathbf{r}_1^T & \mathbf{r}_2^T & r_{13} & r_{33} & 1 & -e_y\mathbf{r}_1^T & -e_y\mathbf{r}_2^T & -e_y r_{13} & -e_x r_{33} \end{bmatrix} \begin{bmatrix} \theta_{14} \\ \vdots \\ \theta_{20} \\ \theta_{22} \\ \theta_{26} \\ \theta_{27} \\ \vdots \\ \theta_{33} \\ \theta_{35} \end{bmatrix} \tag{3.22}$$

Equations 3.21 and 3.22 is expressed in the matrix form named the visuomotor function for a pan-tilt unit

$$\mathbf{e} = V_{PTU} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ r_{13} \\ r_{33} \\ 1 \end{bmatrix} = \begin{bmatrix} v_{PTU1}(\boldsymbol{\theta}) - e_x v_{PTU3}(\boldsymbol{\theta}) \\ v_{PTU2}(\boldsymbol{\theta}) - e_y v_{PTU3}(\boldsymbol{\theta}) \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ r_{13} \\ r_{33} \\ 1 \end{bmatrix} \tag{3.23}$$

where the functions v_{PTU1}, v_{PTU2} and v_{PTU3} are defined by

$$v_{PTU1}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_1 & \cdots & \theta_7 & \theta_9 & \theta_{13} \end{bmatrix} \tag{3.24}$$

$$v_{PTU2}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_{14} & \cdots & \theta_{20} & \theta_{22} & \theta_{26} \end{bmatrix} \tag{3.25}$$

$$v_{PTU3}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_{27} & \cdots & \theta_{33} & \theta_{35} & 0 \end{bmatrix} \tag{3.26}$$

3.1.3 Translation

The derivation of the visuomotor function for 3D translations is now covered. For a translation $\mathbf{t} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$ the expression for ${}^S E'_S$ is

$$ {}^S E'_S(\mathbf{t}) = \begin{bmatrix} I_3 & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.27} $$

As for the visuomotor for PTU, an expression of the visuomotor function for translation is derived from Equations 3.7. Inserting 3.27 in 3.7 gives

$$ e_x = \begin{bmatrix} \mathbf{t}^T & 1 & -e_x \mathbf{t}^T \end{bmatrix} \begin{bmatrix} \theta_{10} \\ \vdots \\ \theta_{13} \\ \theta_{36} \\ \theta_{37} \\ \theta_{38} \end{bmatrix} \tag{3.28} $$

$$ e_y = \begin{bmatrix} \mathbf{t}^T & 1 & -e_x \mathbf{t}^T \end{bmatrix} \begin{bmatrix} \theta_{23} \\ \vdots \\ \theta_{26} \\ \theta_{36} \\ \theta_{37} \\ \theta_{38} \end{bmatrix} \tag{3.29} $$

Equations 3.28 and 3.29 express a linear relation between measurements (motion and image coordinates) and parameters environment and the parameters of the sensors. Since the Equations 3.28 and 3.29 are already linear in $\mathbf{t}$ they are expressed in the following matrix form

$$ \mathbf{e} = V_T \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix} = \begin{bmatrix} v_{T1}(\boldsymbol{\theta}) - e_x v_{T3}(\boldsymbol{\theta}) \\ v_{T2}(\boldsymbol{\theta}) - e_y v_{T3}(\boldsymbol{\theta}) \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix} \tag{3.30} $$

where the functions v_{T1}, v_{T2} and v_{T3} are defined by

$$ v_{T1}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_{10} & \cdots & \theta_{13} \end{bmatrix} \tag{3.31} $$

$$ v_{T2}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_{23} & \cdots & \theta_{26} \end{bmatrix} \tag{3.32} $$

$$ v_{T3}(\boldsymbol{\theta}) = \begin{bmatrix} \theta_{36} & \theta_{37} & \theta_{38} & 0 \end{bmatrix} \tag{3.33} $$

3.1.4 Mobile Robot

Typical mobile robots are modeled as planar robots. The model used in this thesis is illustrated in Fig. 3.5. The motion is composed by a rotation around the Z axis by an angle θ_z followed by a translation $\mathbf{t}$ in the $X-Y$ plane. Accordingly, the motion of the robot is defined by the homogeneous transformation

$$ {}^S E_S(\theta_z, \mathbf{t}) = \begin{bmatrix} R_z(\theta_z) & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & 0 & t_x \\ -r_{12} & r_{11} & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.34} $$

Figure 3.5: Geometric model: Camera frames L and R relative to the world frame W.

Substituting ${}^{S}E_{S}(\theta_z, \mathbf{t})$ in 3.7 and simplifying results in

$$e_x = \begin{bmatrix} r_{11} & r_{12} & t_x & t_y & 1 & -e_x r_{11} & -e_x r_{12} & -e_x t_x & -e_x t_y \end{bmatrix} \begin{bmatrix} \theta_1 + \theta_5 \\ \theta_4 - \theta_2 \\ \theta_{10} \\ \theta_{11} \\ \theta_{13} \\ \theta_{27} + \theta_{31} \\ \theta_{30} - \theta_{28} \\ \theta_{36} \\ \theta_{37} \end{bmatrix} \tag{3.35}$$

$$e_y = \begin{bmatrix} r_{11} & r_{12} & t_x & t_y & 1 & -e_y r_{11} & -e_y r_{12} & -e_y t_x & -e_y t_y \end{bmatrix} \begin{bmatrix} \theta_{14} + \theta_{18} \\ \theta_{17} - \theta_{15} \\ \theta_{23} \\ \theta_{24} \\ \theta_{26} \\ \theta_{27} + \theta_{31} \\ \theta_{30} - \theta_{28} \\ \theta_{36} \\ \theta_{37} \end{bmatrix} \tag{3.36}$$

As for the translation, Equations 3.35 and 3.36 express a linear formulation between measurements (motion and image coordinates) and parameters (environment and sensor).

Again, Equations 3.35 and 3.36 can be expressed in the matrix form called the visuomotor function for mobile robots

$$\mathbf{e} = V_M \begin{bmatrix} r_{11} \\ r_{12} \\ t_x \\ t_y \\ 1 \end{bmatrix} = \begin{bmatrix} v_{M1}(\boldsymbol{\nu}) - e_x v_{M3}(\boldsymbol{\nu}) \\ v_{M2}(\boldsymbol{\nu}) - e_y v_{M3}(\boldsymbol{\nu}) \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ t_x \\ t_y \\ 1 \end{bmatrix} \tag{3.37}$$

where the functions v_{M1}, v_{M2} and v_{M3} are defined by

$$v_{M1}(\nu) = \begin{bmatrix} \nu_1 & \cdots & \nu_5 \end{bmatrix} \tag{3.38}$$

$$v_{M2}(\nu) = \begin{bmatrix} \nu_6 & \cdots & \nu_{10} \end{bmatrix} \tag{3.39}$$

$$v_{M3}(\theta) = \begin{bmatrix} \nu_{11} & \cdots & \nu_{14} & 0 \end{bmatrix}. \tag{3.40}$$

The reason for changing the parameters θ to ν is because the combinations of the θ in Equations 3.35 and 3.36.

3.2 Solution to the Visuomotor Function

Equations 3.16, 3.23, 3.30 and 3.37 express the linear relations between variations of image coordinate and the motion of a camera for different tasks. Each matrix V is supported by a vector of visuomotor parameters θ whose dimensionality depends on the task. In Section 3.3, a method will be presented to estimate the vectors on-line. For now, the concern is about solving for the motion of the camera ${}^{S}E'_{S}$ given an error vector $\mathbf{e}$ and a matrix V. Starting with 6 DOF tasks, the motion of the end-effector is given by the solution to Equation 3.16. Given n image-based errors $\mathbf{e}_i$ we obtain the following system of equations

$$\begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_n \end{bmatrix} = \begin{bmatrix} V_1 \\ \vdots \\ V_n \end{bmatrix} \begin{bmatrix} {}^{S}\mathbf{E}'_{S} \\ 1 \end{bmatrix}. \tag{3.41}$$

Provided that $12 < 2n$ equations are available, the elements of the motion ${}^{S}E'_{S}$ can be estimated from a least squares solution. The solution to Equation 3.41, however, does not respect the constraints of $SE(3)$. That is the solution of Equation 3.41 has twelve dimensions, whereas a transformation in $SE(3)$ has only six degrees of freedom. The reason is that the vectors $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ form a $\mathbb{R}^9$ vector instead of forming a 3×3 matrix in $SO(3)$. Hence, Equation 3.41 requires an appropriate representation for the rotation. Unfortunately, no other chart on $SO(3)$ defines a linear transformation. This implies that replacing $\mathbf{r}_1$, $\mathbf{r}_2$ and $\mathbf{r}_3$ with another representation involves non linear least squares. Also, the representation must be chosen with care since it is a topological fact that any three-dimensional chart on $SO(3)$ has singular configurations [145]. For example, ZYZ Euler angles are singular for $Y = 0$ (Gimbal lock), and, consequently, singular at $R = I$. Another example are rotation vectors or unit vectors and angles which are singular for angles that are multiple of 2π. Rotation representations that are singular at $R = I$ have no appeal for a control law that aims at $R = I$. Unit quaternions have the advantage of being a two-to-one mapping everywhere, but they must be normalized to one. Even though a un-normalized quaternion will represent a rotation after being normalized, it is preferable to solve the equations with the constraint because the translation is not affected by the normalization.

After considering the various charts on $SO(3)$, none of them has all the desired properties to solve Equation 3.41. Although they lead to messy equations, Euler angles is perhaps the least constraining representation because it only imposes a constraint on one parameter. Also, using the ZYX Euler angles, the representation is only singular for $Y = \pm\pi/2$, thus no singularity at $R = I$. Finally, successful algorithms for non-linear pose estimation with ZYX Euler angles have been reported [122].

To derive the upcoming non-linear equations, the following rotation matrices are defined in order to estimate the solution to Equation 3.41 in terms of the ZYX Euler angles

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

$$R_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

6 DOF Motion Estimation

For 6 DOF, define the vector of motion parameters as

$$\mathbf{x}_{6DOF} = \begin{bmatrix} \alpha & \beta & \phi & \mathbf{t}^T \end{bmatrix}^T$$

and define the function to be minimized as

$$f_{6DOF}(\boldsymbol{\theta}, \mathbf{x}_{6DOF}) = V \begin{bmatrix} \left[R_z(\alpha) R_y(\beta) R_x(\phi)\right]_{9\times 1} \\ \mathbf{t} \\ 1 \end{bmatrix} - \mathbf{e} = 0 \tag{3.42}$$

where the $[\ ]_{9\times 1}$ operator indicates the stacking of columns into a 9×1 vector. Given a sufficient number of equations, the least squares solution to Equation 3.42 is found by using Levenberg-Marquardt algorithm (LMA) [157]. For the sake of conciseness, the exact formulation of f_{6DOF} and its Jacobian are presented in the Appendix A.1. Once $\mathbf{x}_{6DOF}$ has been estimated, ${}^{S}E'_{S}$ can be determined and passed to a position controller.

PTU Motion Estimation

For PTU, the vector of motion parameters is simply

$$\mathbf{x}_{PTU} = \begin{bmatrix} \alpha & \beta \end{bmatrix}$$

and the target function f_{PTU} for the parameters $\boldsymbol{\theta}_x$ is given by

$$f_{PTU}(\boldsymbol{\theta}, \mathbf{x}_{PTU}) = V_{PTU} \begin{bmatrix} \cos(\alpha)\cos(\beta) \\ \sin(\alpha) \\ -\sin(\beta)\cos(\alpha) \\ -\cos(\beta)\sin(\alpha) \\ \cos(\alpha) \\ \sin(\beta)\sin(\alpha) \\ \sin(\beta) \\ \cos(\beta) \\ 1 \end{bmatrix} - \mathbf{e} = 0. \tag{3.43}$$

Again, because of the non linearity of Equation 3.43, LMA is used to estimate the least squares solution.

Translation Motion Estimation

Because the translation motion does not involve rotation, the motion is estimated from linear least square. Given n vectors $\boldsymbol{\theta}_T$ and a vector $\mathbf{e}$, the translation parameters

$$\mathbf{x}_T = \mathbf{t}$$

are estimated by solving the linear least squares problem

$$V_T \begin{bmatrix} \mathbf{t} \\ 1 \end{bmatrix} = \mathbf{e}. \tag{3.44}$$

Mobile Robot Motion Estimation

The case of mobile robot motion is similar to the 6DOF and the PTU cases in that it involves a rotation. Using the motion parameters

$$\mathbf{x}_M = \begin{bmatrix} \alpha & t_x & t_y \end{bmatrix},$$

the target function becomes

$$f_M(\boldsymbol{\theta}, \mathbf{x}_M) = V_M \begin{bmatrix} \cos(\alpha) \\ \sin(\alpha) \\ t_x \\ t_y \\ 1 \end{bmatrix} - \mathbf{e} = 0. \tag{3.45}$$

which is again solved by using LMA.

3.3 Estimation of the Visuomotor Parameters

Section 3.1 derived the visuomotor functions for different task spaces. The function for a translational task (Equations 3.28 and 3.29) and the function for a planar task (Equations 3.35 and 3.36) were formulated as a system of linear equations. The function for a pan-tilt task (Equations 3.21 and 3.22) and the function for a 6 DOF task (Equations 3.13 and 3.14), however, were expressed with bilinear terms. Nevertheless, it is possible to express the bilinear terms with a linear term as derived in Equations 3.16 and 3.23.

To compute the solutions presented in Section 3.2, the matrix V must be evaluated and, thus, the visuomotor parameters $\boldsymbol{\theta}$ that compose V must be known. These parameters are estimated on-line from pairs of measurements $({}^S\mathbf{E}_S, \mathbf{e})$ by using an incremental least-squares algorithm [149].

First, Equation 3.16 is manipulated to obtain

$$\mathbf{e} = \Xi\boldsymbol{\theta} = \begin{bmatrix} {}^S\mathbf{E}'_S & 1 & 0 & 0 & e_x {}^S\mathbf{E}'_S \\ 0 & 0 & {}^S\mathbf{E}'_S & 1 & e_y {}^S\mathbf{E}'_S \end{bmatrix} \boldsymbol{\theta} \tag{3.46}$$

Given an overdetermined system of equations $\Xi\boldsymbol{\theta} = \mathbf{e}$, the QR factorization of the $M \times N$ matrix Ξ, with $N < M$, is given by

$$\Xi = QR = Q \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

where Q is a $M \times M$ orthogonal matrix and R is a $M \times N$ matrix composed of the $N \times N$ upper triangular matrix R_1. The least squares solution $\boldsymbol{\theta}_{LS}$ is given by

$$Q^T\mathbf{e} = \mathbf{z} = R\boldsymbol{\theta}_{LS}.$$

A similar result is obtained by considering the augmented system $[\Xi \quad \mathbf{e}]$

$$[\Xi \quad \mathbf{e}] = [Q_1 \quad \mathbf{q}_{N+1}] \begin{bmatrix} R_1 & \mathbf{z} \\ 0 & \rho \end{bmatrix} \tag{3.47}$$

where $\rho = \|\mathbf{e} - \Xi\boldsymbol{\theta}_{LS}\|_2$ is the minimum residual and Q_1 is obtained from the thin QR factorization $\Xi = Q_1 R_1$. The least squares solution to Equation 3.47 is found by solving $R_1\boldsymbol{\theta}_{LS} = \mathbf{z}$ [81].

The rank-k update is done by adding the $k \times N$ matrix Ξ_i to Ξ and noting that

$$\widehat{\Xi} = \begin{bmatrix} \Xi_i \\ \Xi \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix} \begin{bmatrix} \Xi_i \\ R \end{bmatrix}.$$

At this point, all that needs to be done is to transform $\begin{bmatrix} \Xi_i \\ R \end{bmatrix}$ to an upper triangular matrix. By applying an orthogonal transformation Q_i we get

$$\widehat{\Xi} = \begin{bmatrix} I & 0 \\ 0 & Q \end{bmatrix} Q_i^T Q_i \begin{bmatrix} \Xi_i \\ R \end{bmatrix} = \widehat{Q}\widetilde{R} = \widehat{Q} \begin{bmatrix} \widehat{R}_1 & \hat{\mathbf{z}} \\ 0 & \hat{\rho} \end{bmatrix}. \tag{3.48}$$

and the new least squares solution is found by solving $\widehat{R}_1\boldsymbol{\theta}_{LS} = \hat{\mathbf{z}}$.

In the case of rank-1 update, that is $\Xi_i = \xi_i$, the matrix $\begin{bmatrix} \xi_i \\ R \end{bmatrix}$ is Hessenberg upper triangular and is triangulated by a succession of $N+1$ Givens rotations

$$Q_i \begin{bmatrix} \xi_i \\ R \end{bmatrix} = G_N \ldots G_1 \begin{bmatrix} \xi_i \\ R \end{bmatrix} = \widehat{R}$$

which requires $3 * (N+1)^2$ flops. Furthermore, since the new solution only depends on $\widehat{R}$, the computation of $\widehat{Q}$ is not necessary and only a $(N+1) \times (N+1)$ matrix must be allocated to accommodate $\widehat{R}$.

3.3.1 Decoupling of the Visuomotor Parameters

The visuomotor parameters in Equation 3.18 are the result of the coupling between the homogeneous coordinates of ${}^S\mathbf{P}$ and the parameters of the visuomotor cameras $o_{i,j}$. This coupling can be removed by applying a rank-1 approximation to the parameters. For example, noting that

$$\begin{bmatrix} {}^SX \\ {}^SY \\ {}^SZ \\ 1 \end{bmatrix} \begin{bmatrix} o_{31} & o_{32} & o_{33} \end{bmatrix} = \begin{bmatrix} \theta_{27} & \theta_{28} & \theta_{29} \\ \theta_{30} & \theta_{31} & \theta_{32} \\ \theta_{33} & \theta_{34} & \theta_{35} \\ \theta_{36} & \theta_{37} & \theta_{38} \end{bmatrix} = A,$$

the singular value decomposition (SVD) of the matrix A is used to compute a rank-1 approximation. That is, given the SVD decomposition $A = U\Sigma V^T$, the rank-1 approximation is obtained from

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$$

where σ_1 is the largest singular value, $\mathbf{u}_1$ is the first column of the 4×4 matrix U and $\mathbf{v}_1^T$ is the first row of the 3×3 matrix V^T. Given that the coordinates of ${}^S\mathbf{P}$ are homogeneous, we obtain the following result

$${}^S\mathbf{P} = \mathbf{u}_1 / \mathbf{u}_{14}$$

where $\mathbf{u}_{14}$ represents the 4^{th} element of $\mathbf{u}_1$ and the vector $\begin{bmatrix} o_{31} & o_{32} & o_{33} \end{bmatrix}$ is obtained from

$$\begin{bmatrix} o_{31} & o_{32} & o_{33} \end{bmatrix} = \mathbf{v}_1^T \sigma_1 \mathbf{u}_{14}.$$

From there, the other o_{ij} can be recovered as well.

3.4 Summary

This chapter introduced the concept of the visuomotor function. Contrary to other methods, the visuomotor function has the characteristic of implementing feed-forward control by mapping variations of measured image coordinates to variations in task space. Thus, the visuomotor function uses the range of IBVS with the domain of look-then-move. The interaction between the variations in the sensor space and the variations in the task space is captured by a set of parameters whose dimensionality depends on the task. Four task spaces were presented in this thesis: pan-tilt of a robotic head, translation, planar motion and general 6DOF task. Given an error vector $\mathbf{e}$ and the visuomotor parameters for a point ${}^S\mathbf{P}$, the solution of the visuomotor function corresponds to the motion generating the vector $\mathbf{e}$. These visuomotor parameters are estimated on-line from training data by using a recursive least-squares algorithm. In the next chapter, a method will be presented to generalize the visuomotor parameters to 3D volumes around ${}^S\mathbf{P}$.

Chapter 4

Generalization and Function Approximation

Section 3.1 introduced several visuomotor functions as a method to estimate the motion in task space from variations of image coordinates. Section 3.3 presented on-line methods to estimate the parameters of these functions.

From Equations 3.18, the visuomotor parameters depends on the coordinates ${}^{S}X$, ${}^{S}Y$ and ${}^{S}Z$ of a point. It follows that any visuomotor parameters is only valid for a specific 3D point and, consequently, every 3D point has a vector of visuomotor parameters. By itself, it is tempting to perceive this as a caveat of the visuomotor camera. Indeed, the calibration of passive camera is often perceived as a requirement that should be avoided. But given that projective cameras are passive, each projection of a point generates few equations. This is supported by the argument that 3D reconstruction is more accurate given a large amount of well conditioned views of a target. The accuracy of the solution comes from the additional equations that are generated from the different views. Yet, with a passive camera, there is no mechanism to store permanently observations about ${}^{S}\mathbf{P}$ or about the interaction of the camera with ${}^{S}\mathbf{P}$. That is, the parameters of a passive camera only contain information about the sensor itself and these parameters have no relationship to the environment.

On the other hand, given that the visuomotor parameters depend on the coordinates of ${}^{S}\mathbf{P}$, the visuomotor function can store the "characteristics" of ${}^{S}\mathbf{P}$ that were obtained from hundreds or thousands of interactions. In the case of the visuomotor function, these characteristics are the variations of ${}^{S}\mathbf{P}$ projections given a displacement of the end-effector. Thus, the visuomotor parameters allows to store information about ${}^{S}\mathbf{P}$ and that information can be obtained from several views.

Following this, let define a function $\Theta({}^{S}\mathbf{P}) = \theta$ that maps the 3D coordinates of a point to a vector of visuomotor parameters

$$\Theta : \mathbb{P} \rightarrow \mathbb{R}^n$$

where $\mathbb{P}$ represents the 3D coordinates of points that are projected *in the field of view of the camera* and n is the dimension of the vector of visuomotor parameters. The dimensionality of the parameter space depends on the task considered in Chapter 3. One drawback of using a single projection is that the Euclidean coordinates of ${}^{S}\mathbf{P}$ can only be determined up to a scale. By adding a second camera, however, it is well known that the coordinates of ${}^{S}\mathbf{P}$ can be estimated from 3D reconstruction [181]. Instead of aiming to estimate the coordinates of ${}^{S}\mathbf{P}$, let the coordinates of ${}^{S}\mathbf{P}$ be encoded by the stereo coordinates $\mathbf{s}$ by the function

$$\mathcal{S} : \mathbb{P} \rightarrow \mathbb{S}$$

where $\mathbb{S} \subseteq \mathbb{N}^4$ represent the space of stereo coordinates defined by the Cartesian product of both image spaces

$$\mathbb{S} = \mathbb{I} \times \mathbb{I}, \tag{4.1}$$

with the image spaces defined by

$$\mathbb{I} = \{(x, y) \in \mathbb{N}^2 | 0 \leq x \leq N, 0 \leq y \leq M\}$$

where M is the height and N the width in pixels of an image.

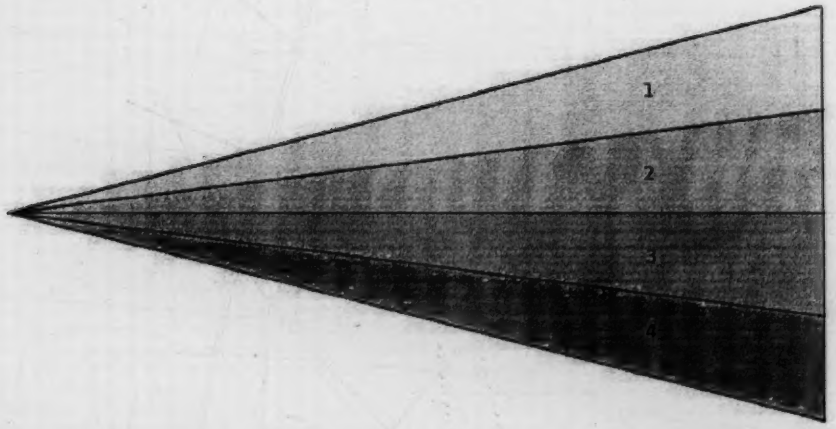


Figure 4.1: Two 1×4 cameras with identical intrinsic and extrinsic parameters. The shaded areas delimit the 4 possible stereo codes.

The cardinality of $\mathbf{S}$, denoted by $|\mathbf{S}|$, determines the total number of stereo codes that are available to encode 3D coordinates and is given by

$$|\mathbf{S}| = (MN)(MN)$$

Let ${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$ denote the 3D space that is visible from both cameras simultaneously and define the corresponding subset of stereo points as $\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$. Essentially, $\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$ represents the subset of stereo points that are obtained from 3D points that are within the field of view of both cameras.

To understand the relation Θ between the space $\mathbb{P}$ and $\mathbb{R}^n$, this chapter analyzes the composition of the function Θ with the function $\mathcal{S}$. The result is a function $\Theta_{\mathcal{S}} \equiv \Theta \circ \mathcal{S} : \mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}} \rightarrow \mathbb{R}^n$, that maps stereo coordinates to the parameter space. First, an upper bound on the cardinality of the domain $\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$ is presented. This bound is used as a guide to determine the scale of the memory requirement for approximating $\Theta_{\mathcal{S}}$ from input-output data. Second, a method is presented to condense the domain $\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$ in order to approximate $\Theta_{\mathcal{S}}$ by a function $\widehat{\Theta}_{\mathcal{S}}$

4.1 Upper Bound on $|\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}|$

To derive a bound on $|\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}|$, two important characteristics of $\mathcal{S}$ are presented. First, because of the discretization done by a CCD array, several points in ${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$ project to a single stereo point in $\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}$. So $\mathcal{S}$ is non injective and this property implies that the stereo coordinates of a 3D point are also associated with other 3D points. These points are confined within the volume that is carved by the intersections of frustums from both cameras. Second, because of the relative position and orientation of the cameras, most stereo codes in $\mathbf{S}$ are not associated with any point in ${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$. That is, most codes provided by the stereo cameras are never observed in practice and are effectively wasted. Thus, in general, the function $\mathcal{S}({}^{S}\mathbf{P})$ is also non surjective.

The nature of the non injectiveness and the non surjectiveness of $\mathcal{S}$ is demonstrated in the following examples. Define two 1×4 line cameras with identical intrinsic parameters. From Equation 4.1, $|\mathbf{S}| = 4 \times 4 = 16$ and, thus, 16 stereo codes are available to encode the volume ${}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}$. Now, assume that the extrinsic parameters are also identical, such that ${}^{L}\mathbf{p}_i = {}^{R}\mathbf{p}_i$ for any ${}^{S}\mathbf{P}$. As illustrated in Figure 4.1, this configuration is clearly non surjective because only 4 stereo codes are possible ($|\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}| = 4$) and 75% of the stereo codes are unused.

This means that, although the stereo cameras provide 16 different coordinates, only 4 of them can be used. Also, the function is non injective because all the points within a shaded area share the same stereo code.

If one of the cameras is slightly translated, $\mathcal{S}$ becomes "less injective" and "less surjective". Figure 4.2 shows that a small lateral translation of a camera yields $|\mathbf{S}_{{}^{L}\mathbb{P} \cap {}^{R}\mathbb{P}}| = 10$ and 37.5% of

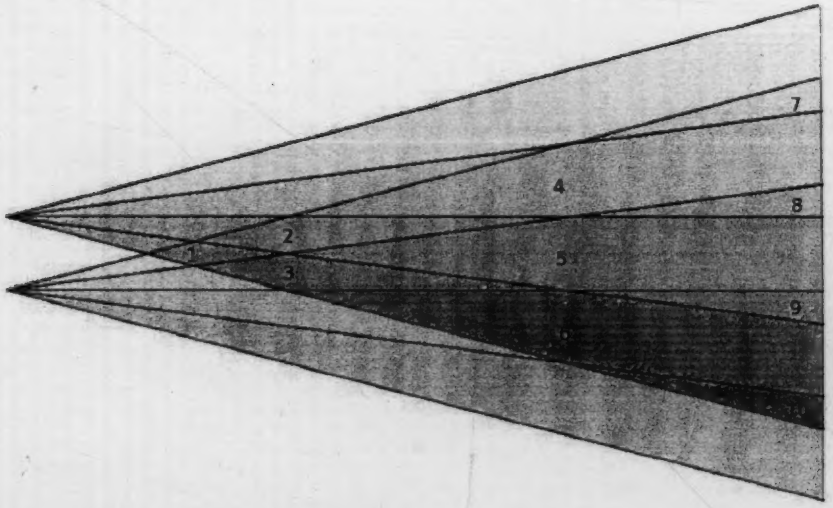


Figure 4.2: Two 1×4 cameras with a translation between their coordinate frames. The regions $1, \ldots, 10$ represent the 10 possible stereo codes.

stereo codes are unused. As a last example, the configuration of Figure 4.3 shows that the function $\mathcal{S}$ is now surjective as $|\mathbf{S}_{^L\mathbf{P} \cap {}^R\mathbf{P}}| = 16$ and all stereo codes are used. Nevertheless, the function is still non injective.

Although surjectiveness is possible in previous 2D example, it is impossible to have two $M \times N$ cameras, with $1 < M$ and $1 < N$, that are configured such that the frustum of every ${}^R\mathbf{p}_i$ intersects the frustums of all ${}^L\mathbf{p}_i$. From epipolar geometry [89], the coordinates of ${}^L\mathbf{p}_i$ are constrained by the epipolar line ${}^L\mathbf{e}$ defined by ${}^R\mathbf{p}_i$ as illustrated in Fig. 4.4. The epipolar line ${}^L\mathbf{e}$ is defined as the projection in the left camera of the ray that connects ${}^S\mathbf{P}$ to the origin of the right camera. Therefore, the length in pixels of the longest epipolar line is used to bound the maximum number of stereo codes that is generated by any epipolar line. Given that the longest epipolar line in a $M \times N$ rectangular image is the image diagonal and that for any image point ${}^R\mathbf{p}_i$, ${}^L\mathbf{p}_i$ is constrained to an epipolar line, then the number of observable stereo codes is bounded by

$$|\mathbf{S}_{^L\mathbf{P} \cap {}^R\mathbf{P}}| < MN\sqrt{M^2 + N^2}.$$

From the previous equations, it is possible to bound the number of individual parameter vectors required to represent any visuomotor function. For example, two 320×240 cameras have at most $(320)(240)\sqrt{320^2 + 240^2} = 30720000$ stereo points to represents any 3D space, which represents only 0.52% of all the possible $(320^2)(240^2)$ stereo points. Therefore, the visuomotor function must store at most 30720000 vectors of visuomotor parameters. For the special case of registered images with horizontal epipolar lines, a tighter bound is given by $24.576e^6$.

Although large, memory allocation of this magnitude falls within the capabilities of today's computers. A more practical concern, however, is the amount of data necessary to estimate each set of parameters. Since each vector $\boldsymbol{\theta}$ involves a least squares solution, the problem of accumulating enough equations to estimate each of them is impractical. Furthermore, it is possible that the parameters associated with a stereo point are not available yet, but that the parameters of a neighboring stereo point are. In such cases, it should be possible to interpolate the neighboring parameters and estimate the unknown parameters. To solve this problem, a sensible approach is to generalize the parameters of each stereo coordinates to neighboring stereo coordinates.

This solution suggests computing an approximation $\widehat{\Theta}_{\mathcal{S}}$ by using a sparse representation of the domain $\mathbf{S}_{^L\mathbf{P} \cap {}^R\mathbf{P}}$. A strategy to address this problem involves methods often used in clustering and pattern recognition. The following sections will discuss about instance-based learning and review the main algorithms to approximate $\Theta_{\mathcal{S}}$ from training data.

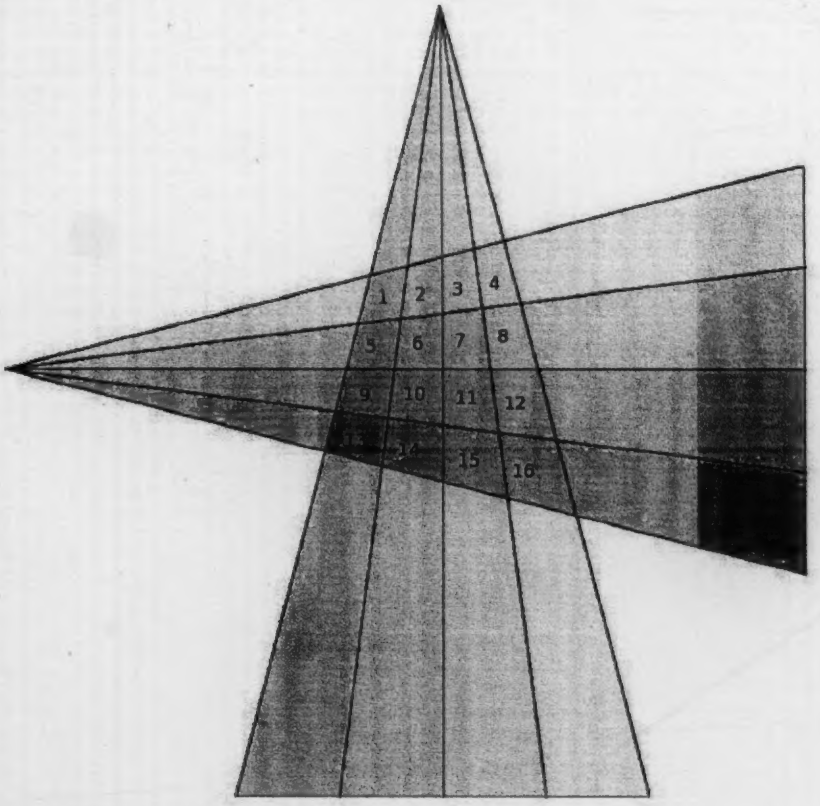


Figure 4.3: Two 1 × 4 cameras oriented in a way to use all 16 stereo codes.

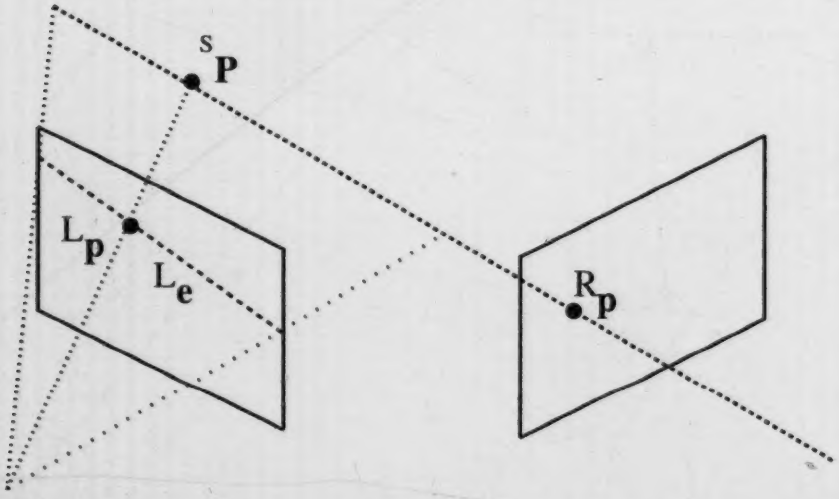


Figure 4.4: Epipolar geometry. The point ${}^{S}P$ projects to ${}^{L}\mathbf{p}$ and ${}^{R}\mathbf{p}$. The epipolar line ${}^{L}\mathbf{e}$ constrains the coordinates of ${}^{L}\mathbf{p}$.

4.2 Instance-Based Learning

Instance-based learning (IBL) methods form a category of algorithms concerned with query time classification. IBL aims at approximating the function $\boldsymbol{\theta} = \Theta_S(\mathbf{s})$ from N pairs $(\mathbf{s}_k, \boldsymbol{\theta}_k)$, $1 \leq k \leq N$, of training data. A characteristic of IBL algorithms is that the classification of the training data is performed at query time. It follows that the approximation $\hat{\boldsymbol{\theta}} = \hat{\Theta}_S(\mathbf{s})$ can be constructed incrementally as new data pairs $(\mathbf{s}_k, \boldsymbol{\theta}_k)$ become available. In its simplest form, IBL involves accumulating raw data and each query is matched directly to a database. In a more elaborate form, IBL scatters basis functions throughout the domain of the function and each query results in the interpolated output of the functions.

The following sections will outline the main IBL algorithms. Over the years, these algorithms have been subjected to several modifications and improvements that are often aimed at specific applications. Within the context of approximating Θ_S, the criteria that are sought are scalability to large data sets, computational efficiency and accuracy of the approximation.

4.2.1 K-Nearest Neighbors

K-nearest neighbors (KNN) is perhaps one of the most straightforward methods to approximate a function from sparse data [144]. The intuition is that each training pair $(\mathbf{s}_k, \boldsymbol{\theta}_k)$ is represented by an entry in a database. Given a query point $\mathbf{s}_q$, the distances between $\mathbf{s}_q$ and each $\mathbf{s}_k$ are evaluated and the algorithm returns the K points with the shortest distance. When $K = 1$, the algorithm returns the closest neighbor and the algorithm is equivalent to a Voronoi diagram [52].

Several specific data structures and query algorithms have been proposed for KNN. Among them, kd trees [52] recursively divide points along each dimension. kd-trees enable sub-linear exact search and fast approximation search algorithms have also been proposed [20]. A main disadvantage of kd-trees, however, is that adding a new point requires a costly re-balancing of the tree. Also, KNN methods are not suitable for incremental updates as the database is allowed to grow arbitrarily large as N increases. For non-injective functions, KNN algorithms imply that only one mapping will be represented in the database. In the case of $\Theta_S(\mathbf{s})$, several points ${}^S\mathbf{P}$ are mapped to the same stereo code $\mathbf{s}$ and each must be represented.

4.2.2 Locally Weighted Regression

Similarly to KNN, locally weighted regression (LWR) uses the nearest neighbors to approximate $\Theta_S(\mathbf{s})$[14, 15]. Contrary to KNN, however, LWR weights each nearest neighbor and the function f is approximated by fitting a function through nearest neighbors.

In practice, LWR often boils down to evaluating a least squares solution over the K-nearest neighbors of the query point where the squared error is weighted according to the distance between the query point and each nearest neighbor.

Because it approximates a function by considering several points within a neighborhood, LWR provides better generalization than KNN. As with KNN, however, LWR does not address the issue of non-injectiveness as a single instance of the mapping $S : \mathbb{R}^3 \rightarrow \mathbb{S}$ is represented in the database. Also, LWR does not provide any mechanism to handle sets of training data that grow arbitrarily large. Thus, for a database that grows incrementally over a long period of time, re-balancing the underlying data structures periodically is computationally expensive.

4.2.3 Radial Basis Functions

As with LWR, radial basis function (RBF) approximates a function by weighting a set of nearest neighbors [144]. The weight attributed to each neighbor i is computed by a kernel function $K(d(\mathbf{s}_q, \mathbf{s}_i))$, where the function d evaluates the distance between the query point $\mathbf{s}_q$ and the i^{th} neighbor. The approximation $\hat{\boldsymbol{\theta}} = \hat{\Theta}_S(\mathbf{s})$ is computed by

$$\hat{\Theta}_S(\mathbf{s}_q) = w_0 + \sum_{i=1}^{L} w_i K(d(\mathbf{s}_i, \mathbf{s}_q))$$

where the value of each weight $w_0, \ldots, w_L$ is determined from the training data. Effectively, this equation represents a two layer artificial neural network in which the first layer is composed of units evaluating the kernel functions and the second layer performs the combination of filters output.

By far, the most widely used kernel function is the Gaussian function

$$K_i(\mathbf{s}) = \exp\left(-\frac{1}{2}(\mathbf{s}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{s}-\boldsymbol{\mu}_i) \right)$$

with mean $\boldsymbol{\mu}_i$ and covariance matrix Σ_i. It is known that Gaussian kernel enables RBF networks to approximate any function within an arbitrary error [144].

RBF with Gaussian kernels represents a specific case of a Gaussian mixture for which the mean and covariance of each kernel are determined and only the amplitudes w_i must be estimated [139]. If the training set is relatively small, a kernel can be centered at each data point. This has the advantage of fitting the training data exactly but it does not extend to on-line approximation since the number of data points can grow arbitrarily large. Also, the kernel functions can be grouped around clusters in the data set by using the EM algorithm [56]. Alternatively, the kernels can be centered at arbitrary intervals throughout the domain of Θ_S. Thus, for a large set of data, Θ_S can be approximated by a relatively small number of kernels that are adequately distributed. Unlike KNN and LWR, this ensures that a fixed number of kernels are used to approximate Θ_S throughout the entire training. One drawback of RBF networks is that for high dimensional spaces, the number of kernels required to adequately cover the domain can be large. Since each kernel function is defined over a broad support, typically the space of real numbers, the number of weights that must be updated for each training pair $(\mathbf{s}_k, \boldsymbol{\theta}_k)$ can be large and computationally expensive. Similarly, to approximate $\hat{\Theta}_S(\mathbf{s}_q)$, the entire set of kernel functions is evaluated.

4.2.4 Cerebellar Model Articulation Controller

Much like RBF, cerebellar model articulation controller (CMAC) globally approximates a function through a combination of local approximations [2]. The main difference between RBF and CMAC is the spatial support and layout of the kernel functions. Although CMAC can be extended to various spatial support, each filter is usually supported by a hypercube. A convenient representation of CMAC is to consider a set of L piecewise constant basis functions, as illustrated in the 2D example of Fig. 4.5. To facilitate the description CMAC and its notation, this section will depart from the previous ones in that it will aim to approximate a function $y = f(\mathbf{x})$ instead of the usual $\boldsymbol{\theta} = \Theta_S(\mathbf{s})$. Let $y \in \mathbb{R}$ and let $\mathbf{x}_k \in \mathbb{R}^2$. Furthermore, define L binary basis functions $\mathbf{b}(\mathbf{x}) = \begin{bmatrix} b_1(\mathbf{x}) & \dots & b_L(\mathbf{x}) \end{bmatrix}^T$, of which only a fixed number $T \leq L$ are activated according to

$$b_l(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ activates the } l^{\text{th}} \text{ basis function} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Then, given a weight vector $\mathbf{w} = \begin{bmatrix} w_1 & \dots & w_L \end{bmatrix}^T$ and a query point $\mathbf{x}_q$, the approximation of $\hat{f}(\mathbf{x}_q)$ is given by the piecewise linear interpolations

$$\widehat{y} = \mathbf{b}^T(\mathbf{x}_q)\mathbf{w} \tag{4.3}$$

Thus, given the basis functions $\mathbf{b}(\mathbf{x})$ and some data $(\mathbf{x}_k, y_k)$, the aim of CMAC is to determine the optimal values of the weight vector $\mathbf{w}$.

Typically, the mapping $\mathbf{b}(\mathbf{x})$ is designed such that two nearby inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ will activate some of the same basis functions (see Fig. 4.5). This is illustrated in Fig. 4.5 where the two neighboring inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ activate the basis function $\mathbf{b}_{L-2}$. Therefore, a methodical arrangement of the receptive fields consists of overlays (also called tilings) such as the one presented in Fig. 4.6(a) [176]. Each overlay is composed of a number of adjacent, non overlapping receptive cells that cover the entire input space. To ensure that each input is covered by exactly T receptive cells, T overlays are superposed with a relative offset such as the example of Fig. 4.6(b) for two tilings.

In the context of the approximating Θ_S. $\mathbf{x} \in \mathbb{R}^2$ must be replaced by the stereo space $\mathbf{s} \in \mathbf{S}$ and the output y is replaced by one of the visuomotor parameters θ_i $(1 \leq i \leq 36)$. Also, each basis function is assigned a set of weight vectors $\mathbf{w}_i$, one for each visuomotor parameter.

Modeling Capabilities

Although CMAC was developed in 1970s, no thorough analysis of CMAC was investigated until 1989 [154]. To understand the limitations of CMAC one must study the type of function that CMAC is capable of modeling and the convergence of the algorithm. Originally, it was reported that CMAC

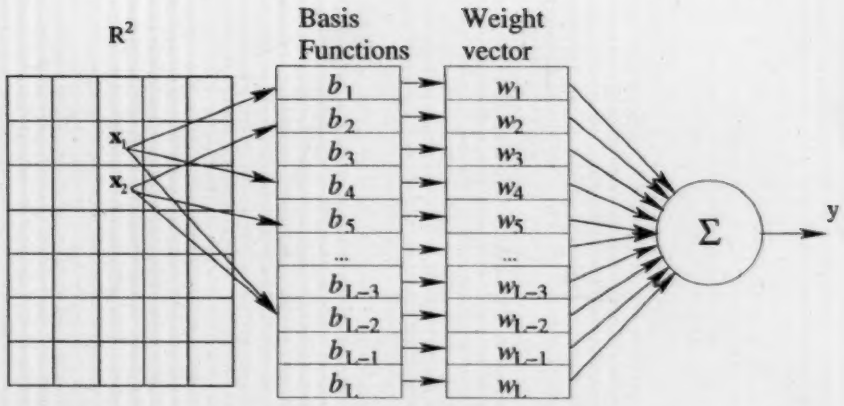
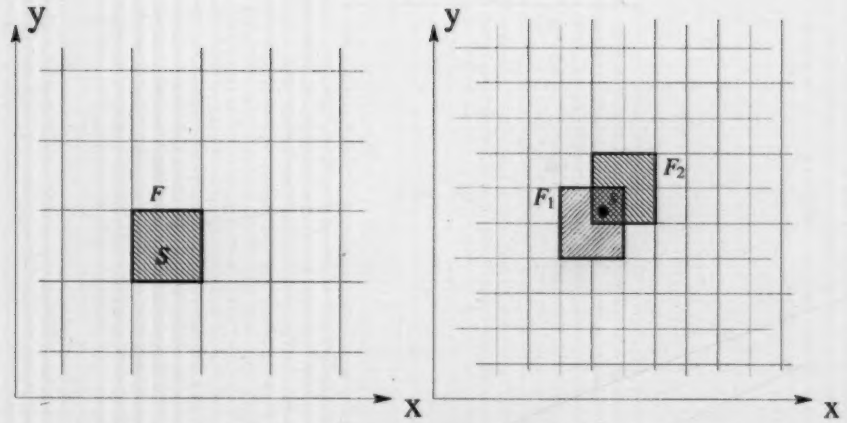


Figure 4.5: Example of a 2D CMAC. Each input $\mathbf{x} \in \mathbb{R}^2$ activates $T = 3$ basis functions. Each basis function b_l is associated to a weight w_l that is combined to approximate y.



(a) Single overlay of a 2D state space.

(b) Superposition of two overlays ($T = 2$).

Figure 4.6: Positioning of receptive fields.

always converge with arbitrary accuracy on univariate and multivariate inputs [66]. Later, this claim was shown to be erroneous as CMAC cannot reproduce arbitrary multivariate look-up tables [28]. The same research also concludes, however, that CMAC is able to model exactly functions that are the linear combinations of univariate constant functions. The authors demonstrate their claim by proving that the dimension of the set of L basis function of a CMAC network is $L-(T-1)$. This important result implies that when the basis functions overlap each other ($1 < T$), the space spanned by the basis functions is of lesser dimension than the number of basis functions L. As a consequence, given N pairs of input-output $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ with $L <= N$, the $N \times L$ matrix

$$B = \begin{bmatrix} \mathbf{b}(\mathbf{x}_1)^T \\ \cdots \\ \mathbf{b}(\mathbf{x}_N)^T \end{bmatrix} \tag{4.4}$$

used to solve the system of equations

$$B\mathbf{w} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \mathbf{y} \tag{4.5}$$

has rank$(B) \leq L-(T-1)$. As such, when more than one overlay is used ($1 < T$), rank$(B) < L$ and the matrix B is singular. The generalization also affects adversely the dimension of the range of B. The conclusion from these findings is that there is a space of multivariate functions that have no CMAC equivalent and this space increases as T increases.

Brown *et al.* demonstrate, however, that CMAC is able to model functions that are defined by a linear combination of piecewise constant functions. In detail, they demonstrate how a function $f(\mathbf{x}) = \sum_i f_i(x_i)$, where each $f_i(x_i)$ is a piecewise constant function, can have an equivalent CMAC representation.

Convergence

The convergence properties of CMAC have been studied and debated [27, 29, 154, 155, 66, 65]. Because CMAC uses piecewise constant interpolants, the output is also piecewise constant. Thus, CMAC can only model exactly piecewise constant functions. The formulation of CMAC by piecewise constant interpolants suggest formulating the problem as an optimization over a set of basis functions. Early research thought that the basis functions are always linearly independent and, thus, the linear optimization was done on a full rank system [66, 65]. Later on, as mentioned in the previous section, it was demonstrated that linearly independence of the basis functions is only a property of univariate CMAC and that the basis functions of multivariate CMAC are always linearly dependent, resulting in an optimization problem on a rank deficient system [29].

Parks and Militzer [154] were among the first to provide formal answers about the convergence of the original CMAC algorithm presented by Albus in [2]. Their research shows that CMAC either converges to a fixed point or a "limit cycle" when the data are presented in "cyclic training". For "random training", they conjecture that CMAC converges to "capture zones". Parks and Militzer also compared several learning algorithms to train a CMAC network. Namely, those are

1. Albus-Kaczmarz learning algorithm.
2. Moving average training.
3. Partially optimized step length in the last perpendicular direction.
4. Training at the point with maximum error.
5. Partial Gram-Schmidt.

Although all these methods aim at solving Equation 4.5, they are formulated as recursive algorithms. The reason for incremental algorithm is essentially related to the computational cost of solving Equation 4.5 directly. Noting that B is an $N \times L$ matrix and N is often in the order of $N \backsim 10000$ while $L \backsim 1000$, solving Equation 4.5 through the Penrose-More or singular value decomposition (SVD) is a challenge for real-time systems. The conclusions are that despite partial Gram-Schmidt is able to approximate the functions exceptionally well, it only does so in the absence of noise and it is computationally expensive. In comparison, the Albus-Kaczmarz algorithm has a median convergence rate but has the cheapest computational cost.

In other research, the convergence of a least mean square algorithm is presented in [26]. He *et al.* present a LMS algorithm with variable learning rate, but assumed that B is full rank. Similarly to the recursive least squares algorithm presented in Section 3.3, an algorithm based on the normal equation [158] and on the QR factorization [159] were presented.

4.3 Approximation of Θ_S

CMAC and RBF are effective on-line approximation algorithms [175] that have comparable performances [112]. Because of the convenient shapes of each cell, however, CMAC with Albus-Kaczmarz, has the advantage of being computationally very efficient [174]. As will be outlined in Chapter 5, the computational efficiency of CMAC, enables several updates of parameters at each iteration by queuing the updates.

The two important components of CMAC are the computation of the basis vector $\mathbf{b}(\mathbf{s})$ and the algorithm used to update the weight vector $\mathbf{w}$. In this thesis, the basis vector is obtained from a set of hashing functions while the Albus-Kaczmarz algorithm is used to update the weights.

4.3.1 Vector of Basis Functions

Using CMAC, Θ_S is approximated by L 4D hypercubes organized in T overlays. In practice, the mapping $\mathbf{b}(\mathbf{s}_q)$ is done by a set of T hashing functions $h_j : \mathbb{S}_{L\mathbb{P}\cap R\mathbb{P}} \rightarrow \mathbb{R}^L$ with $1 \leq j \leq T$, where each h_j is only allowed to activate one basis function on the j^{th} overlay. Hence, for T overlays, a query point $\mathbf{s}_q \in \mathbb{S}_{L\mathbb{P}\cap R\mathbb{P}}$ is mapped to T basis functions according to

$$\mathbf{b}(\mathbf{s}_q) = \sum_{j=1}^{T} h_j(\mathbf{s}_q). \tag{4.6}$$

Once determined, the basis functions are used to approximate the parameters by summing the weight vectors

$$\hat{\boldsymbol{\theta}} = \widehat{\Theta}_S(\mathbf{s}_q) = \begin{bmatrix} \mathbf{b}(\mathbf{s}_q)\mathbf{w}_1 \\ \cdots \\ \mathbf{b}(\mathbf{s}_q)\mathbf{w}_n \end{bmatrix} \tag{4.7}$$

where $\mathbf{w}_i$, $1 \leq i \leq 38$ indicates the weight vector associated with the visuomotor parameter $\boldsymbol{\theta}_i$. The stacking of $\mathbf{b}(\mathbf{s}_q)\mathbf{w}_i$ is necessary since the CMAC assumes by default a single output. Thus, 38 weight vectors are necessary and each visuomotor parameter is associated to a $\mathbf{w}_i$.

4.3.2 Albus-Kaczmarz Learning Algorithm

The original algorithm proposed by Albus [2], aimed at using the training data $(\mathbf{s}_k, \boldsymbol{\theta}_k)$ to incrementally adjust the vector $\mathbf{w}$ to obtain an accurate approximation. The algorithm itself has its roots in Rosenblatt's Perceptron [161] and even further to Hebbian theory [93].

Given N equations $(\mathbf{s}_k, \boldsymbol{\theta}_k)$, the goal of the algorithm is to find suitable values $\mathbf{w}$ to satisfy, if possible, Equation 4.3. In the case of approximating Θ_S, the vectors $\boldsymbol{\theta}_k$ are obtained from the incremental least squares (Section 3.3). In vector form, the vector $\mathbf{w}_i$ after $k+1$ updates, denoted by $\mathbf{w}_{i,k+1}$ is given by

$$\mathbf{w}_{i,k+1} = \mathbf{w}_{i,k} + (\theta_{i,k+1} - \mathbf{b}(\mathbf{s}_{k+1})^T \mathbf{w}_{i,k}) \frac{\mathbf{b}(\mathbf{s}_{k+1})}{T}, \tag{4.8}$$

where $\theta_{i,k+1}$ represents the $k+1^{\text{th}}$ estimate of the i^{th} visuomotor parameter. Essentially, Equation 4.8 adds the correction $\theta_{i,k+1} - \mathbf{b}(\mathbf{s}_{k+1})^T \mathbf{w}_{i,k}$ to the current weights. Then, correction is distributed uniformly on the T weights that have contributed to the error by multiplying with $\frac{\mathbf{b}(\mathbf{s}_{k+1})}{T}$.

It was later found that this method is identical to the one presented by Kaczmarz in 1937, known as the Kaczmarz method [105]. Originally, the iterative method was designed to solve systems of linear equations and has been used in several areas such as signal processing, machine learning and computer tomography where it is known under the name Algebraic Reconstruction Technique (ART) [147]. Kaczmarz also reported that the distribution vector $\frac{\mathbf{b}(\mathbf{s}_{k+1})}{T}$ can be multiplied by a factor $0 < \epsilon < 2$ and still preserve convergence. An identical finding was also reported in [92].

Perhaps the most significant problem with Albus-Kaczmarz method is that the convergence depends on the order in which the equations are presented. Parks and Militzer clearly demonstrate this by analyzing the convergence of the algorithm according if the data is presented in cyclic order "cyclic training" or if the in random order "random training" [154]. Although better convergence has been reported with "random training" [95, 69, 147], no theoretical derivation of this convergence has been demonstrated other than the conjecture in [154]. Recently, however, Stromher and Vershynin have proposed a randomized Kaczmarz algorithm with exponential decay [173]. Their algorithm

selects an equation with a probability that is determined by its relevance. Although Stromher's algorithm fills a large gap in Kaczmarz algorithms, the method is not suitable for on line approximation because the algorithm only selects specifics equations with the implication that all the equations are initially available.

Chapter 5

Experiments

Experiments were conducted to assess the performance of the visuomotor function. This chapter involves experiments in simulations and with real robots. The aim of these experiments is to evaluate the performance of the visuomotor function for each task presented in Chapter 3.

As mentioned in Chapter 4, stereo cameras are used to index the visuomotor parameters by using an approximation of the function $\hat{\Theta}_S$. These parameters, however, only capture the image-based variation of a single camera. That is, even though two cameras are used, the visuomotor parameters represent a single visuomotor camera. This implies that the second camera is used only to obtain a stereo point and it is not used to solve for the motion of the end-effector. It is possible, however, to obtain a vision motor function for both camera. In this case, two functions are approximated simultaneously (one function for the left and one function for the right) and, thus, two sets of parameters are associated with each stereo point: one set of parameters ${}^{L}\boldsymbol{\theta}$ is estimated for the left camera and one set of parameters ${}^{R}\boldsymbol{\theta}$ is estimated for the right camera. The difference between these two approaches is that in the latter case, since the domain of both functions is the motion of the end-effector, both functions can be combined to solve for the motion of the end-effector. Essentially, this implies that using two functions generates twice as many equations than using a single function. For example, in the 6DOF case, at least six equations are required to find a solution to the motion of the end-effector (Section 3.2). Since each image point provides two constraints, three points are required to find a solution. If a single function is used, then the error of three points must be defined in the corresponding camera. If two functions are used, however, the equations can be distributed in both function and, thus, both cameras. In the previous example, two equations can come from one camera and four from the other. If the image-based error of three points is defined in both cameras, then a total of 12 equations are available to solve for the motion of the end-effector.

Furthermore, using two functions instead of one does not constrains a point to be visible from both cameras in the initial and in the desired positions. As long as a 3D point provides a stereo correspondence in the initial view and an image-based error is defined in either of the cameras, the point provides two constraints to the solution. In the case where the number of initial stereo correspondences is insufficient, it is also possible to use the visuomotor parameters given by the desired views and to solve for the inverse of the motion. Although it is possible to use equations obtained from the initial and final views simultaneously, it must be remembered that the extra equations are redundant.

In the following experiments, every simulation was done by using a single visuomotor function (associated with the left camera) while experiments performed on a real robot used two functions except experiments with a real PTU.

5.1 Implementation Details

Details about the implementation of the algorithms presented in Chapters 3 and 4 are now given. Simulations were implemented in Matlab while experiments on a real robot were implemented in C++.

Incremental Least Squares

The incremental least squares algorithm based on the QR factorization can be easily implemented in C++ by linking to LAPACK and BLAS libraries. For each set of parameters, a sufficiently large

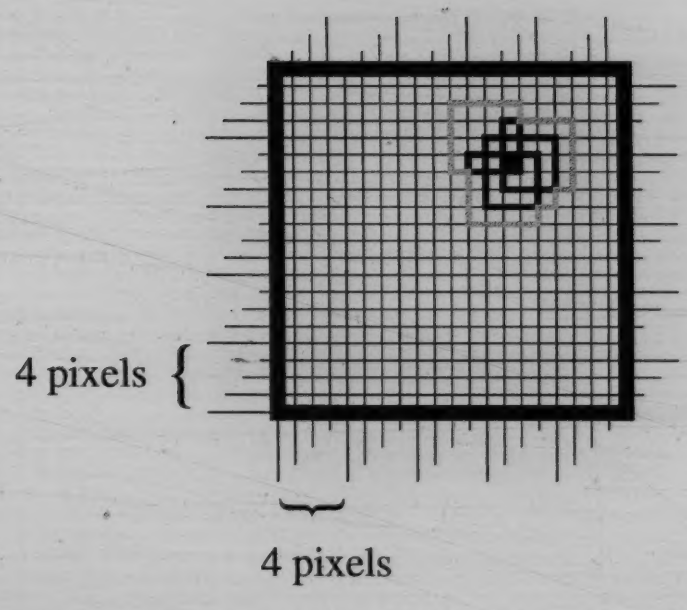


Figure 5.1: Overlays for 2D images. Each color represent an overlay and each bold contour represents the selected tile on each overlay. The yellow contour represent the boundary from the union of all the selected tiles.

matrix must be maintained to store the upper triangular matrix R and k update equations. In the C++ implementation, the QR factorization is done by calling the LAPACK routine *dgeqrf* and the solution to the triangular system is computed by *dtrtrs*. As mentioned in Section 3.3, it is possible to formulate the problem with an upper Hessenberg matrix and to factorize by using a sequence of Givens rotations. Although this is computationally very efficient it is limited to rank-1 updates. For the simulations in Matlab, the QR factorization routine was simply called.

CMAC

Each visuomotor function is approximated by representing the stereo space $\mathbf{S} \subset \mathbf{N}^4$ with a CMAC. In the experiments, each image coordinate is encoded by 32 overlays with each overlay dissecting the image in a grid of 32×32 pixels. These values were chosen because they provide a resolution of 1 pixel and a reasonable amount of generalization. This can be verified with the example of Figure 5.1 with 4 grids of 4×4 pixels. The figure shows that by offsetting each grid by exactly one pixel along both x and y directions, each pixel is encoded by a unique combination of 4 tiles. Although the total area over which the parameters are generalized varies between image coordinates (see the yellow contour and the black dot in Figure 5.1) the 32×32 tiles provide a reasonable trade-off between generalization, accuracy and memory.

In the example of Figure 5.1, the image delimited by the thick black square contains $20 \times 20 = 400$ pixels. Yet, each overlay contains $5 \times 5 = 25$ tiles such that a total of 100 tiles are required for an approximation. Thus, for one 320×240 image and 32 overlays of 32×32 pixels, each of the 76,800 image coordinates is encoded by a unique combination of 32 tiles among 2,560 tiles. For 640×480 images, each of the 307,200 pixels are encoded by the combination of 32 tiles among 9,600 tiles.

Levenberg-Marquardt

In the C++ implementation, the solution to the motion of the end-effector was computed by using the *lmder* routine in the MINPACK package [1]. In the Matlab implementation, the routine *LMFsolve*

[1] MINPACK is available from www.netlib.org.

available on the internet was used[2].

Updates Pipeline

Approximating the visuomotor function involves updating several vectors of visuomotor parameters. The characteristic of these parameters is that they capture the interaction between the camera and a specific stereo point. The fact that each visuomotor parameter depends on the 3D coordinates of a point (see Equation 3.7) provides the capability to store information about that interaction for each point. The main drawback of the visuomotor approximation is that these parameters must be estimated. Because each tile contains a set of parameters, the total number of tiles (i.e. 2,560) is an upper bound on the number of sets of parameters that must be estimated. It is important to note, nevertheless, that some of these tiles might never be used. This claim is supported by the discussion of section 4.1 that demonstrates that not all 3D point can be represented by a stereo point. Because the relation between the 3D space and the stereo space depends on the configuration of the cameras, it is possible for a 3D point to be visible by only one camera.

By itself, the estimation of the parameters with incremental least squares and the approximation with CMAC do not pose a problem. These algorithms are stable and divergence is exceptional. The main challenge stems from the amount of data that is required to estimate each set of parameters.

The motivation for the visuomotor approach is that as the end-effector moves, the cameras track or match features in the stereo space and the observed variations are used to estimate the visuomotor parameters. The algorithm presented in this section collects these variations in a list and updates the parameters of all the tiles that are represented in the list. Although the algorithm can be extended to several points, it only requires that a single 3D target is tracked in both cameras in order to generate a sequence of stereo points $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_n$.

For example, lets assume that the initial position and orientation of the end-effector with respect to the base frame is ${}^{S_1}E_B \in SE(3)$ and let $\mathbf{s}_1$ be the stereo coordinates of the target in that position. Now, let the robot move to a new position defined by ${}^{S_2}E_B$ according to the relative transformation ${}^{S_2}E_{S_1}$. During the motion, the target is tracked and its coordinates are now $\mathbf{s}_2$. From ${}^{L}x_2 - {}^{L}x_1$, ${}^{L}y_2 - {}^{L}y_1$ and ${}^{S_2}E_{S_1}$, the parameters $\Theta_S(\mathbf{s}_1) = \boldsymbol{\theta}_{\mathbf{s}_1}$ are updated. The update procedure consists of updating the least squares estimate of $\boldsymbol{\theta}_{\mathbf{s}_1}$ (Section 3.3) followed by the updating the weights vectors in the CMAC (Equation 4.8). Given the same data, it is also possible to update the parameters $\Theta_S(\mathbf{s}_2) = \boldsymbol{\theta}_{\mathbf{s}_2}$. This comes from the observation that ${}^{L}x_1 - {}^{L}x_2$, ${}^{L}y_1 - {}^{L}y_2$ and ${}^{S_1}E_{S_2}$ can be obtained from $\mathbf{s}_1$, $\mathbf{s}_2$ and ${}^{S_2}E_{S_1}$

Assume that the robot keeps moving and the end-effector is at position ${}^{S_3}E_W$ and the target is has the coordinates $\mathbf{s}_3$. Then, from ${}^{L}x_3 - {}^{L}x_1$, ${}^{L}y_3 - {}^{L}y_1$ and ${}^{S_3}E_{S_1}$ the parameters $\boldsymbol{\theta}_{\mathbf{s}_1}$ are updated once again. Using the above argument, the parameters $\Theta_S(\mathbf{s}_3) = \boldsymbol{\theta}_{\mathbf{s}_3}$ are also updated from ${}^{L}x_1 - {}^{L}x_3$, ${}^{L}y_1 - {}^{L}y_3$ and ${}^{S_1}E_{S_3}$. Now, if ${}^{S_2}E_{S_1}$ and $\mathbf{s}_2$ are stored, it is also possible to update the parameters $\boldsymbol{\theta}_{\mathbf{s}_2}$ from ${}^{L}x_3 - {}^{L}x_2$, ${}^{L}y_3 - {}^{L}y_2$ and ${}^{S_3}E_{S_2}$. Finally, it is also possible to update once again the parameters $\boldsymbol{\theta}_{\mathbf{s}_3}$ from ${}^{L}x_2 - {}^{L}x_3$, ${}^{L}y_2 - {}^{L}y_3$ and ${}^{S_2}E_{S_3}$.

This procedure can be repeated each time an observation $({}^{S_i}E_{S_1}, \mathbf{s}_i)$ becomes available by using the combinations between pairs of observations as illustrated by Figure 5.2. Thus, if a list contains N pairs of observations $({}^{S_j}E_1, \mathbf{s}_j)$ the parameters $\boldsymbol{\theta}_{\mathbf{s}_i}$ can be updated $N-1$ times from the other observations in the list. Repeating the procedure for each node in the list yields a total of $N(N-1)$ updates. Consequently, each time a new observation $({}^{S_i}E_1, \mathbf{s}_i)$ is obtained, the parameters $\boldsymbol{\theta}_{\mathbf{s}_i}$ are updated N times from all the $({}^{S_j}E_1, \mathbf{s}_j)$ in the list. Likewise, each observation in the list can be updated once from $({}^{S_i}E_1, \mathbf{s}_i)$. The pipeline algorithm is outlined by the Algorithm 1. The routine $UpdateParameters$ first updates the parameters with the incremental least squares and, then, the new estimate is propagated in the CMAC (Equation 4.8).

As it will be demonstrated in the upcoming experiments, Algorithm 1 is capable of estimating the visuomotor parameters very efficiently. In the experiments, the updates were done incrementally and the initial pipeline only contains the initial observation $({}^{S_1}E_{S_1}, \mathbf{s}_1)$. Then, each new observation $({}^{S_i}E_1, \mathbf{s}_i)$ is used for one round of updates. As the number of observations in the pipeline grows, the number of updates performed during each iteration grows linearly. Realistically, given limited computational resources, the number of observations in the pipeline must be bounded.

Thus, although the argument that the visuomotor function requires the estimation of a large set of parameters is valid, the challenge is addressed by taking advantage of the large amount of data

[2]The file LMFsolve.m is available from the Matlab Central website.

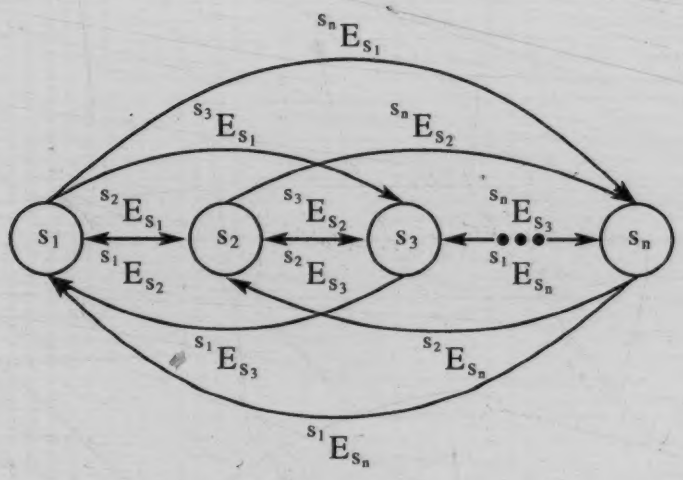


Figure 5.2: Combination for updates.

Algorithm 1 Approximate the visuomotor function

$N \leftarrow 0$
$transformation \leftarrow \emptyset$
$stereo \leftarrow \emptyset$
{Do one round of updates}
loop
  $^{S}E_{W} \leftarrow$ EndEffector() {Obtain the position of the end-effector}
  $\mathbf{s} \leftarrow$ StereoTarget($^{S}E_{W}$, $^{W}\mathbf{P}$) {Obtain the stereo coordinates}
  for $i = 1$ to N **do**
    $^{S_i}E_{W} \leftarrow transformation[i]$
    $\mathbf{s_i} \leftarrow stereo[i]$
    $tiles \leftarrow$ GetTiles($\mathbf{s_i}$) {Update the parameters associated with $\mathbf{s_i}$}
    for $t = 1$ to T **do**
      UpdateParameters($tiles[t]$, $^{S_i}E_{W}$, $^{S}E_{W}$, $\mathbf{s_i}$, $\mathbf{s}$)
    end for
    $tiles \leftarrow$ GetTiles($\mathbf{s}$) {Update the parameters associated with $\mathbf{s}$}
    for $t = 1$ to T **do**
      UpdateParameters($tiles[t]$, $^{S}E_{W}$, $^{S_i}E_{W}$, $\mathbf{s}$, $\mathbf{s_i}$)
    end for
  end for
  $N \leftarrow N + 1$
  $transformation[N] \leftarrow {}^{S}E_{W}$
  $stereo[N] \leftarrow \mathbf{s}$
end loop

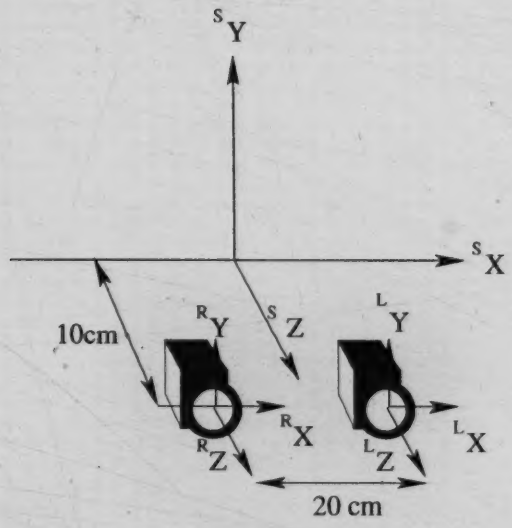


Figure 5.3: Configuration of the PTU used in the simulations.

that is provided by the cameras and the robot.

5.2 Pan-Tilt Unit

PTU are often used in robotics to build robotic heads. Controlling the angles of a PTU from visual feedback, often called "gaze control", is perhaps the simplest task from an IBVS perspective. Since only two degrees of freedom are controlled two equations are required to control the pan and tilt angles. For this task, IBVS requires two equations which can be provided by the image-based error of a single image target. Because IBVS are feedback systems, the amount of pan and tilt involved in a task is never determined ahead or during the execution of the task.

Typical feed-forward systems that can determine the amount of pan and tilt rely on 3D reconstruction. This involves estimating the initial and desired 3D coordinates of the target in the coordinate frame of the end-effector and then solving for the pan and tilt angles.

Using the solution to the visuomotor function, the pan and tilt angles can be determined from the image-based error measured in one camera and the second camera is used only to index the table of parameters. Although using stereo cameras seems redundant to control a PTU, it must be recalled that most PTU do not merely orient the coordinate frame of a camera. That is, virtually every PTU also involve an amount of translation. It is for this reason that the parameters depends on the depth of ${}^{S}\mathbf{P}$. In the case of a true PTU, then no translation is involved and a monocular camera is sufficient.

5.2.1 Simulations: Visuomotor Function

The simulations consist of projecting a 3D point in a virtual camera and generating motions by selecting random pan and tilt angles. Only angles that resulted in the point being projected within the field of view of the camera were retained.

The horizontal baseline between the cameras was set to 20cm and both cameras were moved 10cm in front of the end-effector (Figure 5.3). The intrinsic parameters of the virtual cameras were obtained from a 640×480 Point Grey Research Dragonfly. The parameters of the cameras are reported in Table 5.1. Gaussian noise with a standard deviation of 3 pixels was added to both image coordinates.

Algorithm 1 was implemented and two thousand iterations were executed. The implementation of the routine $EndEffector()$ samples random pan and tilt angles and returns a transformation ${}^{S_i}E_W$. Although the angles were randomly sampled, only the angles that resulted in ${}^{W}\mathbf{P}$ being projected in the field of view of both cameras were selected. The routine $StereoTarget$ returns the stereo coordinates of the point ${}^{W}\mathbf{P}$ when the end-effector is in position ${}^{S}E_W$.

Focal length (horizontal):	469
Focal length (vertical):	472
Central point (horizontal):	344
Central point (vertical):	255

Table 5.1: Intrinsic camera parameters of the camera used in simulation.

At the end of each iteration of Algorithm 1, the solution to the visuomotor function was computed by solving 3.43 with ${}^{L}x_i - {}^{L}x_1$, ${}^{L}y_i - {}^{L}y_1$ and from the parameters $\boldsymbol{\theta}_{\mathbf{s}_1}$. A sample of simulated angles and the solutions obtained during one simulation are presented in Figure 5.4. Figure 5.5 illustrates the absolute errors between the simulated angles and the solutions (the absolute value of each error). The figures show that although the initial solutions are erroneous, the solutions improve as more observations are gathered and updates are processed. The error stabilize after 100 frames, which is equivalent to 200 rank-1 updates.

Because only the parameters of observed stereo point are estimated, an update map is presented in Figure 5.6. The intensity of a pixel indicates how many times the pixel was associated with a tile whose parameters were observed. For example each time that the pixel $({}^{L}x, {}^{L}y)$ was observed, the tiles that are associated with that pixel (see Figure 5.1) were updated. For each tile $1 \leq t \leq 32$ encoding $({}^{L}x, {}^{L}y)$, all the pixels that are masked by $tiles[t]$ are incremented by 1 to indicate that they are associated with a tile that was updated. This process is repeated for all the stereo points $\mathbf{s}_i$. Because it is difficult to visualize the 4D space of stereo points, each image is visualized individually. Although this does represent the combination of left/right pixels inherent to stereo points, the maps provide an insight on how the updates are distributed in each image. One observation about Figure 5.6 is that not all stereo points are observed. As mentioned in Chapter 4, not all stereo points can be observed, thus, not every tile is updated. Another observation is that the random sampling of angles results in a relatively uniform distribution of updates.

This simulation was repeated 100 times and the statistics about the mean absolute errors (Figure 5.5) are presented in Figure 5.7. The plots indicates that the mean absolute error converges to 0.075rad for the pan angles and 0.05rads for the tilt angles with the standard deviation of 0.035rads and 0.02rads respectively.

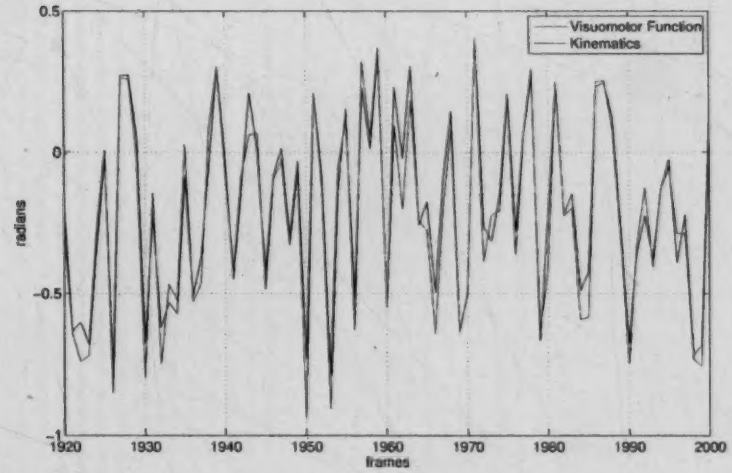
To demonstrate the robustness of the visuomotor approach, 100 more simulations were done. This time again, the parameters used to compute the solution to the function were those obtained from the projection of ${}^{W}\mathbf{P}$. The image-based errors however were obtained by the projection of ${}^{W}\mathbf{P} + \mathcal{N}(0, 0.25m)$ where $\mathcal{N}(0, 0.25m)$ corrupts each coordinate of ${}^{W}\mathbf{P}$ with a Gaussian disturbance of standard deviation 0.25m. In other words, this experiment aims to demonstrate that the parameters associated with a stereo point generalize well to neighboring 3D points. The results are illustrated in Figure 5.8. Compared to Figure 5.7, the disturbances roughly doubles the mean absolute error and the standard deviation.

Real

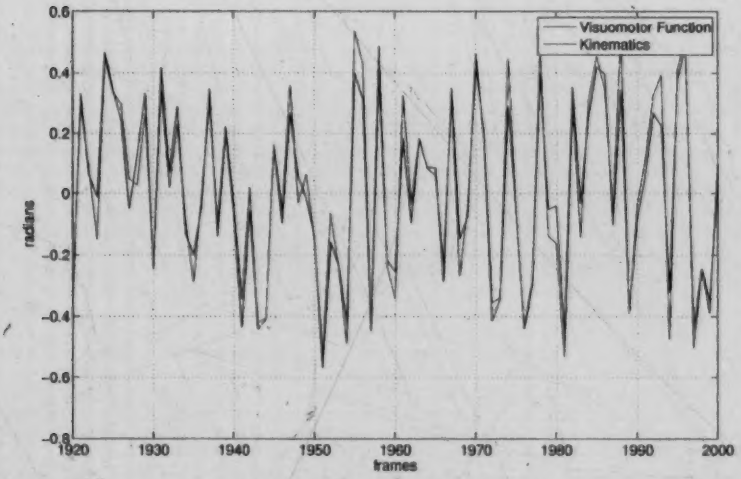
Experiments done with a real PTU involved two video cameras mounted on the wrist of a robot arm. The cameras used in the experiments were two Pyro IEEE 1394 webcams that were positioned approximately 20cm apart as illustrated in Figure 5.9. Each camera captured a 320×240 image in YUV422 format 30 times per second. One target in both images was tracked by the meanshift algorithm [42]. The pan-tilt system used in the following experiments used joints 5 and 6 of a 7 DOF Whole Arm Manipulator (WAM).

The procedure for this experiment follows the same guideline than the simulation with the exception that real data is collected. The meanshift tracker was initialized on a target and the target was tracked throughout the experiment. The back-drivable nature of the WAM enables moving the arm in gravity compensation and, by using this characteristic, the wrist was moved manually. At each frame, the image coordinates of the meanshift tracker and the corresponding position of the end-effector were recorded (see $EndEffector$ and *StereoTarget* in Algorithm 1). The variations derived from the data were used to approximate the visuomotor function of one camera on-line. Algorithm 1 was used on-line with the data streaming from the cameras and the WAM.

To compare the results of the experiments with those obtained in simulations, the solution of the visuomotor function was evaluated after each iteration of Algorithm 1. The pan and tilt angles are displayed with the solutions from the visuomotor function in Figure 5.10 and the corresponding absolute errors are presented in Figure 5.11. Compared to the simulation, the errors stabilize after 400 frames. Beyond the 400th frame the mean absolute errors are 0.0419rad for the pan angles

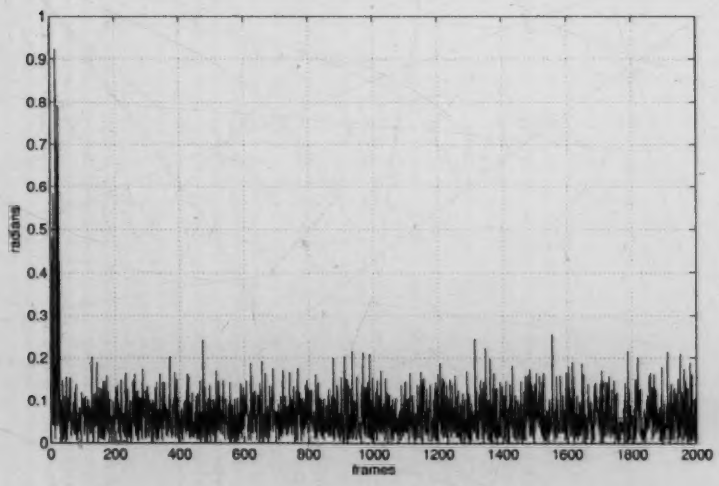


(a) Pan angle (rad).

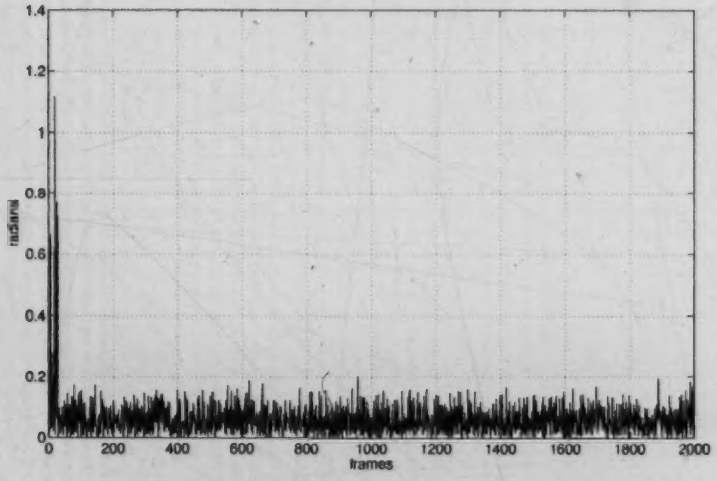


(b) Tilt angle (rad).

Figure 5.4: Actual and estimate values of pan and tilt angles (only the last 80 iterations are illustrated for clarity reason)

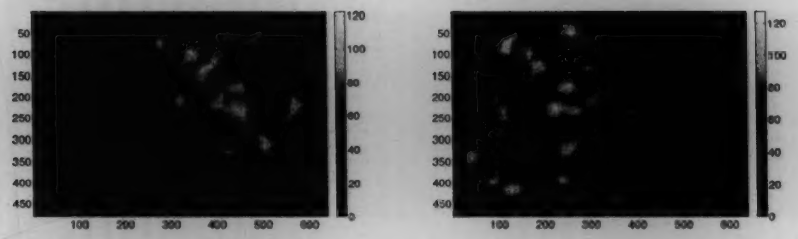


(a) Absolute pan error (rad).



(b) Absolute tilt error (rad).

Figure 5.5: Absolute error of pan and tilt angles.



(a) Simulation PTU: Update map for the left camera. (b) Simulation PTU: Update map for the right camera.

Figure 5.6: Update map. The intensity of a pixel reflects the amount times one of its tile was selected.

and 0.0411rad for the tilt angles with standard deviations of 0.0344 and 0.0310 respectively. One explanation for this is the conditioning of the observations used in simulation. Contrary to random angles, the angles used in this experiments vary smoothly and lead to poorly conditioned systems of equations. Also, by moving the PTU manually, the user tends to keep the target towards the center and generates small image-based errors. This can also be visualized from the update map of Figure 5.12 where the most updated areas are the most intense. The maps clearly demonstrate the bias towards the center of the image. Also, this concentration is a consequence of the configuration of both cameras. With the PTU, it is impossible to move away from the target such that the space $\mathbb{S}_{^L\mathbf{P}\cap^R\mathbf{P}}$ is very limited. Hence, although the parameters of these stereo point get updated frequently, the equations are poorly conditioned.

5.3 Translation

Translational movements are often required to do reaching movements. From an IBVS perspective, translational tasks pose an interesting problem. In hybrid systems, such as $2^1/_2$ visual servoing, the translation component is only determined up to a scale. Therefore, translation motions are controlled from the visual feedback. As with many IBVS tasks, this approach only considers the direction of the translation, but not the amount of translation involved.

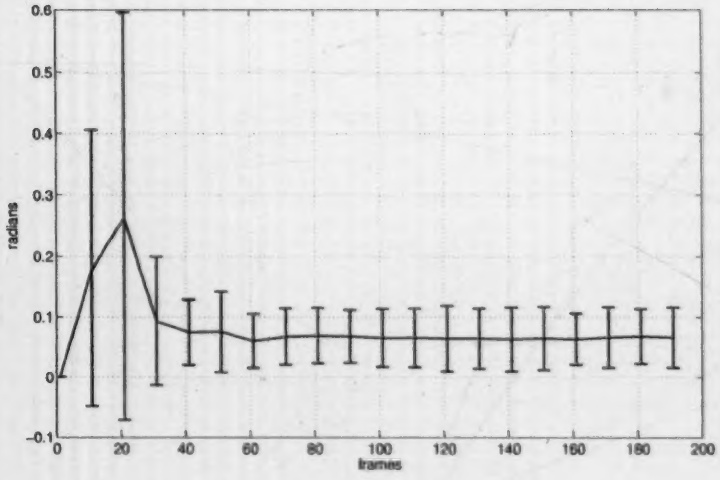
Of course, with calibrated stereo cameras, the translation can be estimated from 3D reconstructions. Given a stereo correspondence in the initial image and a correspondence in the desired image, the 3D coordinates of the point in the initial and desired positions can be estimated and the translation can be recovered.

Contrary to this approach, the visuomotor function uses the parameters associated with each stereo point involved in the task. Based on these parameters and the image-based errors, the visuomotor function is solved directly for the translation. Thus, at least three equations obtained from the image-based errors are required. These equations can come from a combination of sources. If one visuomotor function is used, the solution requires the image-based errors from at least two points (four equations). If two visuomotor functions are used, then the same 3D point provides two equations for each function for a total of four equations (two equations for the function of the left camera and two equations for the function of the right camera).

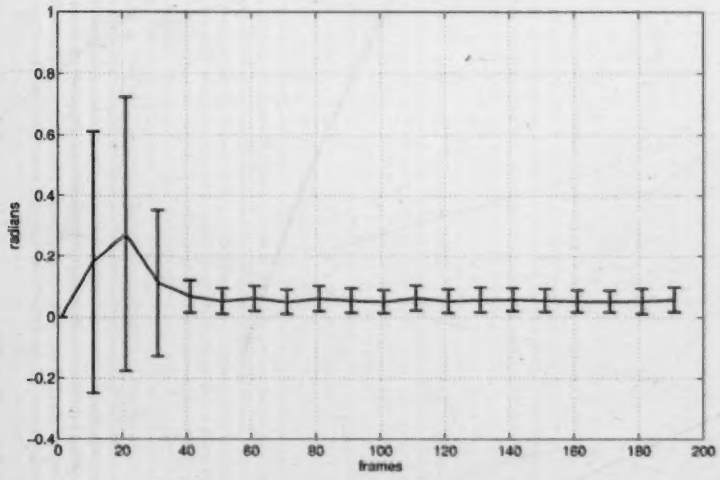
Simulation: Visuomotor Function

The simulations presented in this section are based on those presented in section 5.2.1. The cameras have the same internal and external parameters and the 3D translations were sampled randomly. The translations of the end-effector were confined to the interval $\begin{bmatrix}\pm 1m & \pm 1m & \pm 1m\end{bmatrix}^T$, although only the translations that kept the targets within the field of view of both cameras were used. One visuomotor function was used in the experiments, which implies that two 3D points were used to provide the constraints to the solution.

Each simulation consisted of 500 iterations of Algorithm 1. After the each iteration, the translation between the initial position and the most recent one was estimated by solving Equation 3.44 by using the parameters $\boldsymbol{\theta}_{\mathbf{s}_1}$ and the image-based errors obtained from ${}^Lx_i - {}^Lx_1$ and ${}^Ly_i - {}^Ly_1$. The results of one simulation is illustrated in Figures 5.13, 5.14 and 5.15. Each figure plots the simulated translations (X, Y, Z) along with the solutions to Equation 3.44. The figures also plots the absolute error for each of the solution.

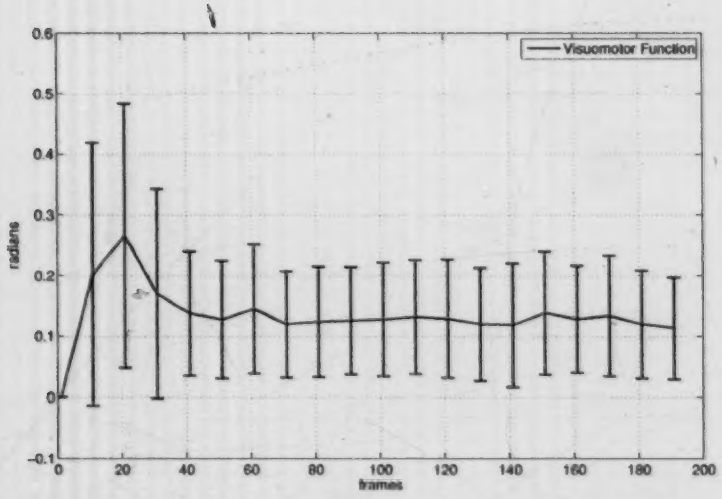


(a) Mean and standard deviation of the absolute pan error.

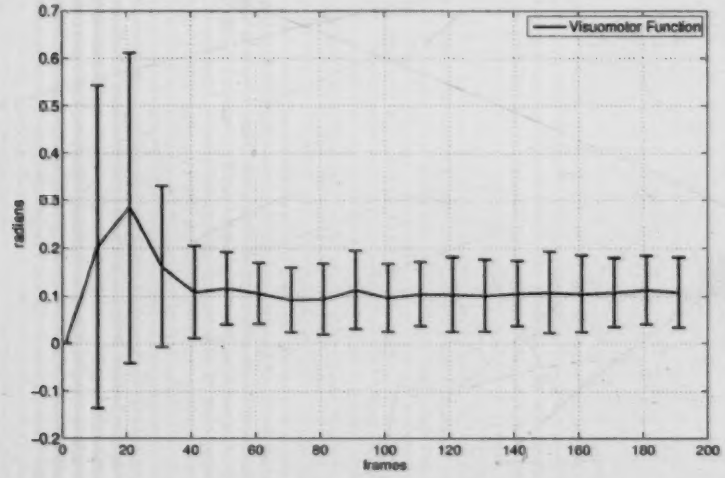


(b) Mean and standard deviation of the absolute tilt error.

Figure 5.7: Error statistics of 100 simulations.



(a) Mean and standard deviation of the absolute pan error after adding normal disturbance to the 3D point.



(b) Mean and standard deviation of the absolute tilt error after adding normal disturbance to the 3D point.

Figure 5.8: Error statistics of 100 simulations.

Figure 5.9: Setup for the eye-in hand cameras used for the PTU and for the 6DOF experiments.

This simulation was performed 100 times and the mean absolute error and its standard deviation are reported in Figure 5.16, 5.17 and 5.18. The reported errors show that the error of the solutions stabilize within 50 frames. The convergence can be verified by monitoring the condition number of the system of equations used to compute the least squares estimate of $\boldsymbol{\theta}_{\mathbf{a}_1}$. This information was recorded during one of the simulations and it is illustrated in Figure 5.19. The figure shows that the condition number stabilizes after 50 frames. Thus, any additional updates beyond the 50th frame has little influence on the estimation of the parameters and, consequently, on computing a solution.

Again, to demonstrated the robustness of the visuomotor approach to disturbances, the system was asked to determined the translations of a point ${}^{W}\mathbf{P} + \mathcal{N}(0, 0.1)$ while using the parameters associated with the stereo point of ${}^{W}\mathbf{P}$. Thus 100 additional simulations were done and the results are shown in Figures 5.20 to 5.22. Clearly, the disturbances have a more negative effects on the results, especially when compared to those of the same experiment with a PTU. As it will be demonstrated in the next section, however, these results still compare favorably to the 3D reconstruction approach.

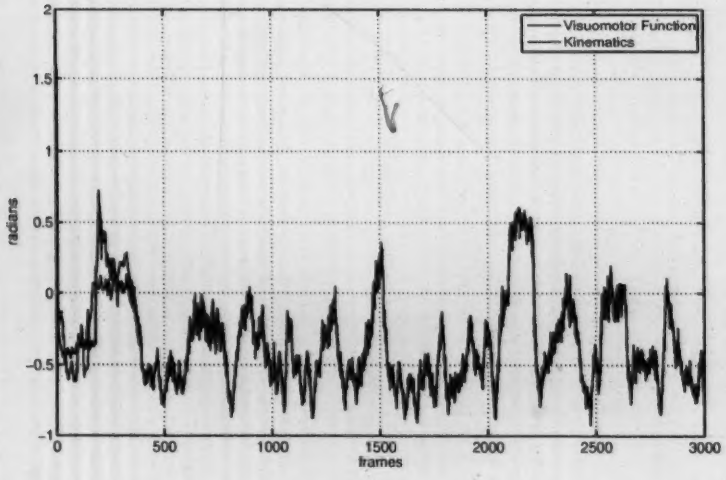
Simulations: 3D Reconstruction

In this section the performance of the simulations presented in the previous section are compared to estimating the translation from 3D reconstruction. The stereo rig used in these simulation was identical to the one used in the previous simulations. One of the 3D point used in the previous simulations was also used in this experiment and the same cameras and noise were used. Furthermore, the *exact* camera parameters were used to perform the 3D reconstruction. That is, the parameters used for the perspective projection were also used for the 3D reconstruction. The algorithm employed for the 3D reconstruction is the maximum likelihood (ML) algorithm that minimizes re-projection errors [89]. This algorithm is considered among the most accurate and robust and the Matlab routines were downloaded from the textbook's website[3].

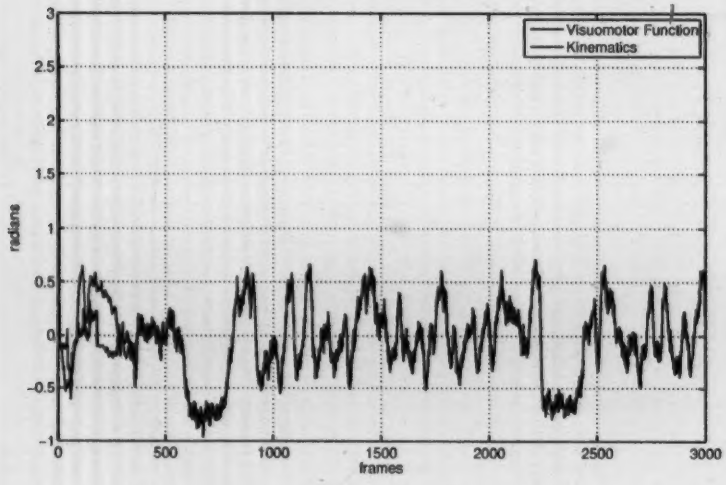
The experiment consists of 100 simulation with each simulation performing 500 translations. The translations were selected according to the guideline presented in the previous section. First, ${}^{W}\mathbf{P}$ was projected in the stereo cameras to the point $\mathbf{s}_1$ and its 3D coordinates were estimated according to the ML algorithm. Then, the end-effector was translated and ${}^{W}\mathbf{P}$ was projected to the point $\mathbf{s}_2$ and the final 3D coordinates were estimated from these projections. Finally, the translation was determined from subtracting the initial 3D coordinates from the final 3D coordinates. The results of the mean absolute errors and standard deviations over all the simulations are illustrated for each component in Figures 5.23 5.24 and 5.25.

As illustrated, the results for the 3D reconstruction are surprisingly bad. For the X and Y components, the mean absolute error is consistent at 0.5m whereas it varies between 0.6m and 0.8m for the z component. The standard deviation averaging 0.45m is also strikingly large. Compared to the visuomotor simulations of Figures 5.16, 5.17 and 5.18, the errors from the 3D reconstruction more than 10 times greater. Yet, both methods used 4 equations to solve the same problem. In fact, the results obtained with 3D reconstructions are comparable to the results obtained in Figure 5.20 to 5.22 where disturbances were added to the environment.

[3]The code is available at http://www.robots.ox.ac.uk/ vgg/hzbook/code. The specific Matlab routine used in the experiments is vgg_X_from_xP_nonlin.

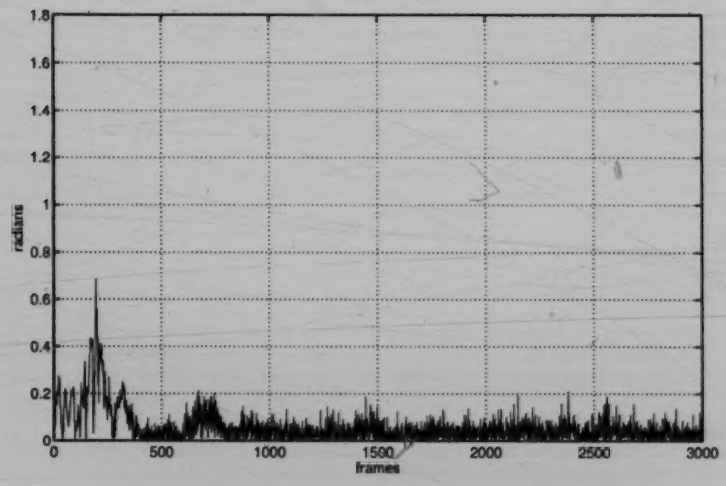


(a) Real and estimated pan angles.

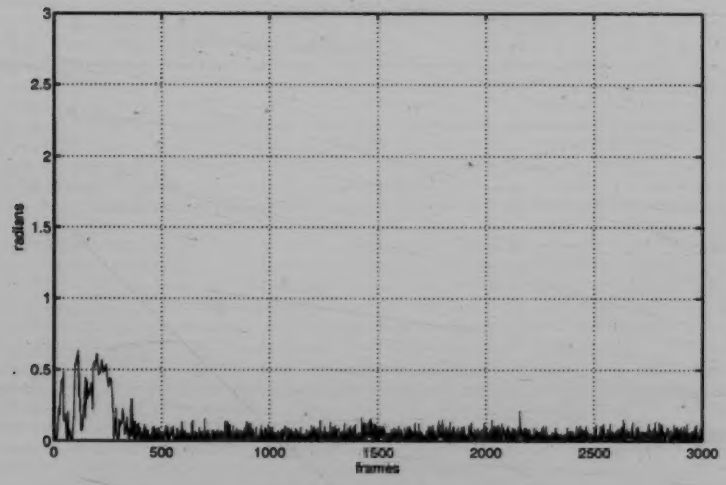


(b) Real and estimated tilt angles.

Figure 5.10: Real PTU: Pan and tilt angles.

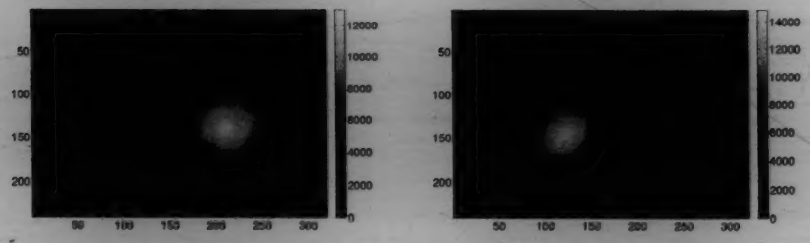


(a) Absolute error for pan angles.



(b) Absolute error for tilt angles.

Figure 5.11: Real PTU: Absolute errors.



(a) Real PTU: Update map for the left camera.

(b) Real PTU: Update map for the right camera.

Figure 5.12: Update map. The intensity of a pixel reflects the amount of updates.

One explanation of this difference is that the 3D reconstruction propagates errors when estimating the initial coordinates and the final coordinates of ${}^{W}\mathbf{P}$. Since the translation is directly derived from these values, the errors from both estimations accumulate in the estimation of the translation. Contrary to this, the visuomotor approach does not transform the problem in the 3D space. Therefore, no error is propagated in the 3D space and the solution only depends on the accuracy of the parameters and the input noise.

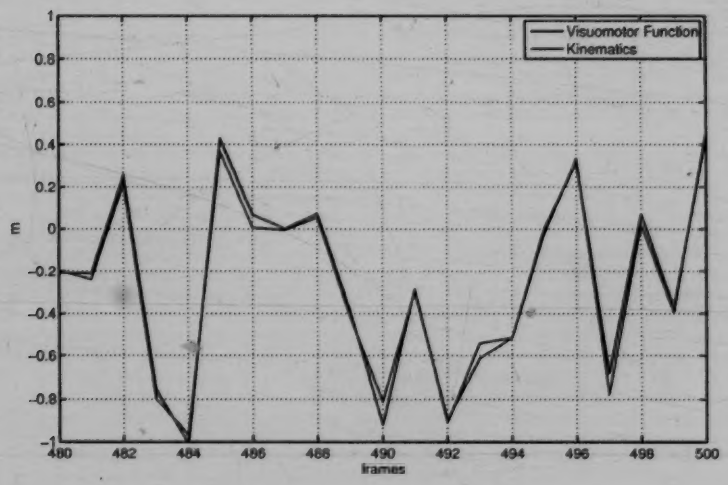
Another explanation is that although the exact parameters of the camera were used for the 3D construction, these parameters only define the behavior of the sensor. Contrary to this, the parameters of the visuomotor camera depend on the sensor and on the projective coordinates of the target and the parameters define the interaction of the camera with the environment. Furthermore, the parameters of the visuomotor camera are estimated by observation resulting from motion in a specific task space. Thus, the parameters associated with a stereo point represent the interaction of the camera with a 3D point, or more correctly of a 3D volume, under the task space. And it is possible to store the parameters of such interactions because the visuomotor function is able to model such interaction. In this sense, the visuomotor function has an unfair advantage over the 3D reconstruction methods. Yet it is the nature of the visuomotor function to dedicate parameters to each stereo point and it is the nature of the algorithms presented in this thesis to use a large amount of data in order to approximate them.

4 DOF WAM

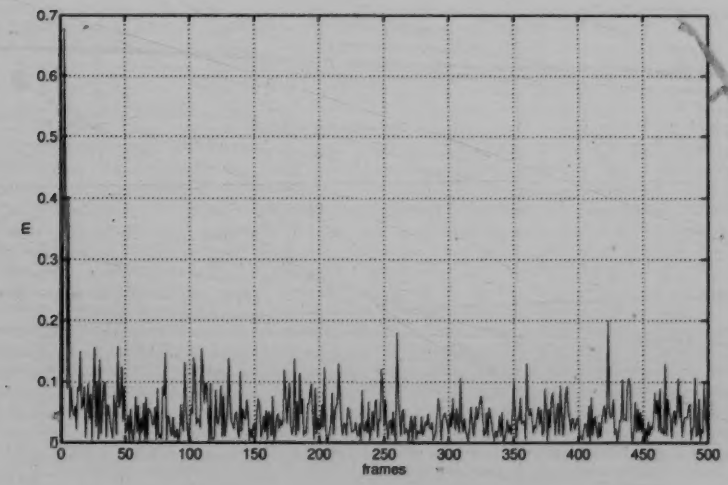
Experiments for translations were performed with a 4 DOF Whole Arm Manipulator (WAM). A marker was installed on the end-effector of the WAM and the marker was tracked by meanshift trackers in both cameras. This standalone configuration was chosen because the (X, Y, Z) coordinates of the end-effector are independent from its orientation. As such, the end-effector is not constrained to translational motions. Thus, the XYZ position of ${}^{W}\mathbf{P}$ is determined by the translation of the end-effector. Because only one 3D point was used and 4 equations are required to solve for the translation, two visuomotor functions were used. That is visuomotor parameters were estimated for the both cameras simultaneously. The solution of the translation was computed by direct least-squares using the Fortran routine *dgelss*.

Using the Algorithm 1, the parameters were updated at the frame rate of the cameras (30fps). The routine $EndEffector$ simply queried the forward kinematics of the arm, while the routine $StereoTarget$ returned the image coordinates of the target. The arm was move manually in gravity compensation for 3000 frames. The results are illustrated in Figures 5.26, 5.27 and 5.28. These figures are the counterparts to the simulation results of Figures 5.13, 5.14 and 5.15.

As for the PTU, the main difference between the simulations and the real system is the convergence horizon. The real system requires about 200 frames for the solutions to stabilize whereas the simulations converged within 50 frames. The mean absolute errors beyond the 500th frame were $[0.0311m \quad 0.0335m \quad 0.1277m]$ with standard deviations of $[0.0250m \quad 0.0252m \quad 0.0998m]$. Again, the nature of the real motion appears main culprit for the slower convergence as illustrated by the evolution of the condition number for the system of equations for one of the tiles associated with $\mathbf{s}_1$ (Figure 5.29. Another possible cause is related to the eye-to-hand configuration used in the experiment. Because the marker on the end-effector has to remain visible to both cameras the volume in which the end-effector was able to move is significantly reduced. One reason is self-occlusion where the target on the end-effector becomes occluded by the end-effector itself. In the simulation experiment,the end-effector was able to move in a $2m \times 2m \times 2m$ cube whereas in the end-effector of the WAM was restricted to a cube of about $0.6m \times 0.8m \times 0.6m$.

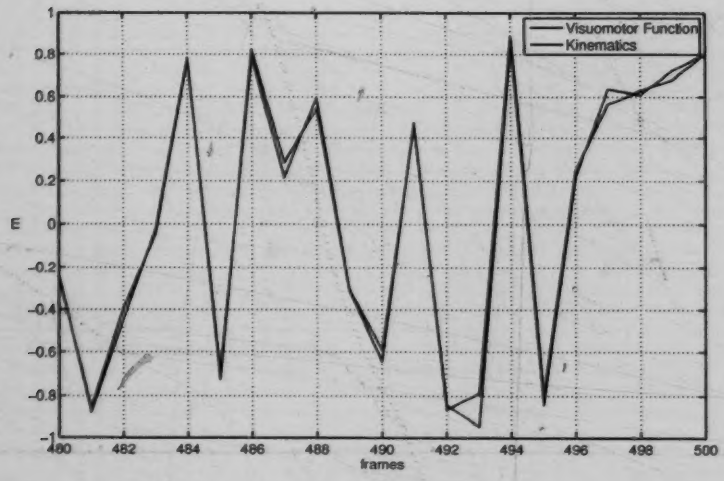


(a) Simulated X translation.

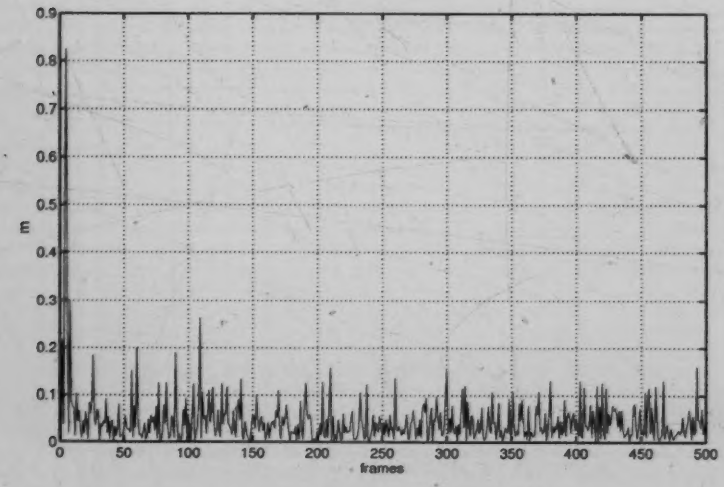


(b) Absolute error of simulated X translation.

Figure 5.13: Simulated translations along the X axis.

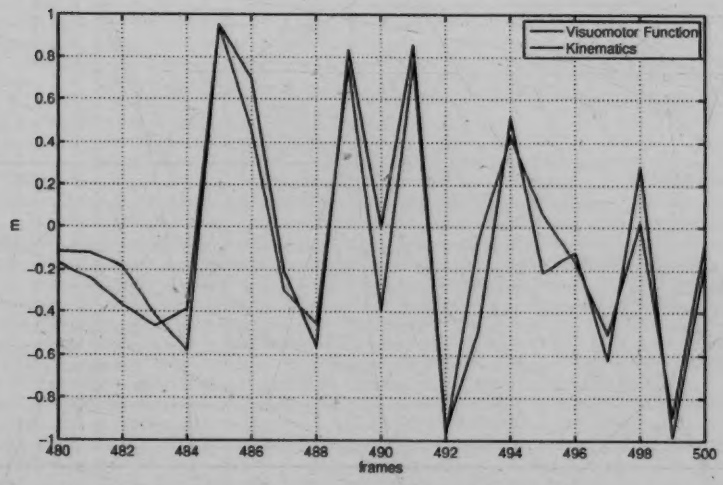


(a) Simulated Y translation.

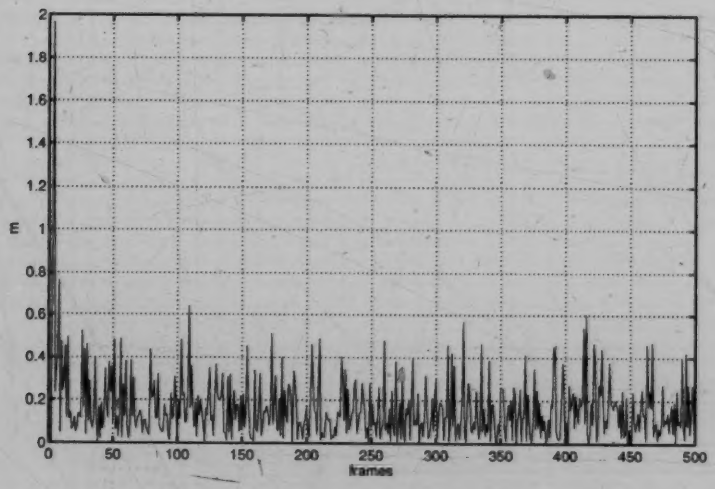


(b) Absolute error of simulated Y translation.

Figure 5.14: Simulated translations along the Y axis.



(a) Simulated Z translation.



(b) Absolute error of simulated Z translation.

Figure 5.15: Simulated translations along the Z axis.

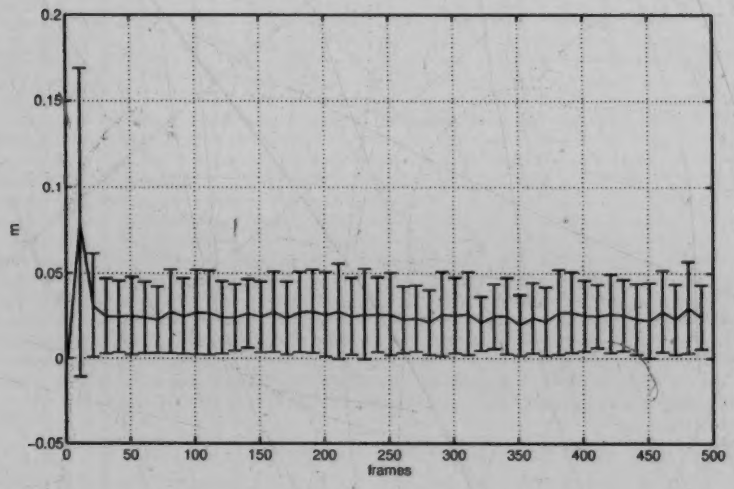


Figure 5.16: Mean and standard deviation of the absolute X error (100 simulations).

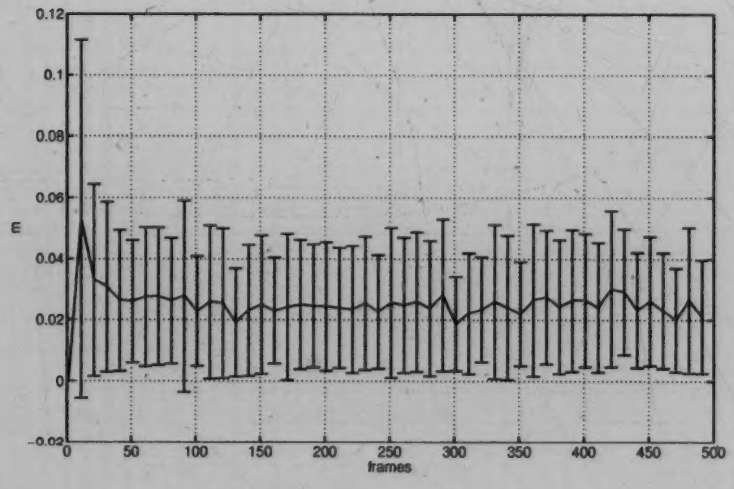


Figure 5.17: Mean and standard deviation of the absolute Y error (100 simulations).

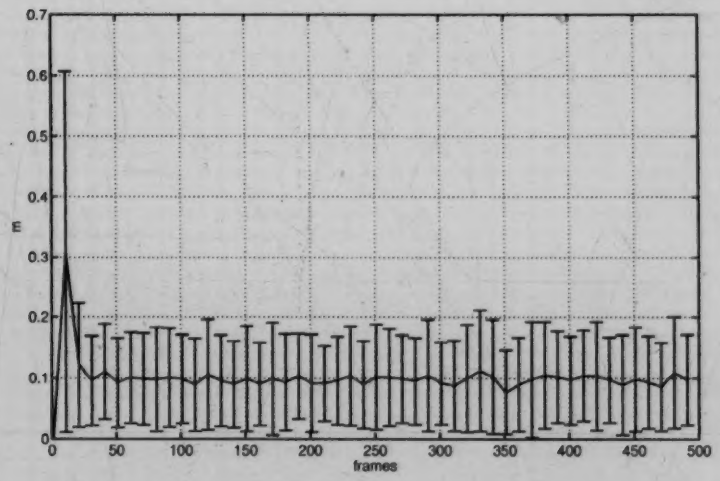


Figure 5.18: Mean and standard deviation of the absolute Z error (100 simulation).

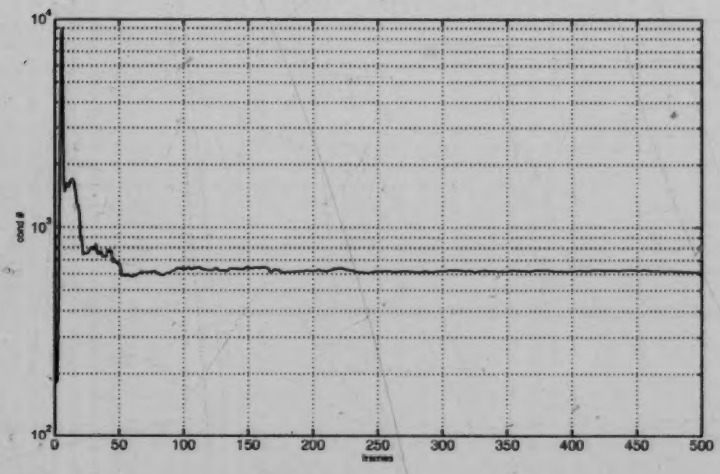


Figure 5.19: Evolution of the condition number of for one tile with simulated data.

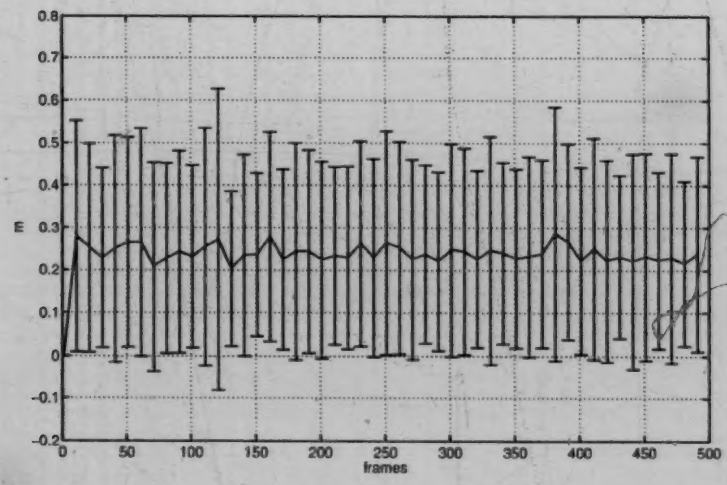


Figure 5.20: Mean and standard deviation of the absolute X error with additional disturbances (100 simulations).

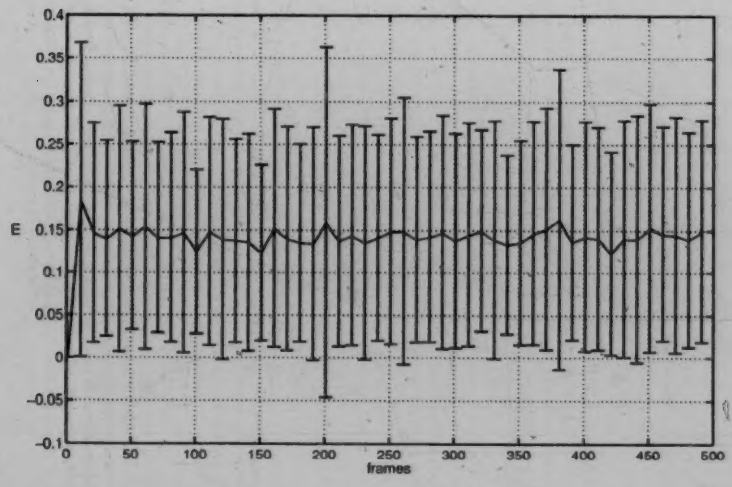


Figure 5.21: Mean and standard deviation of the absolute Y error with additional disturbances (100 simulations).

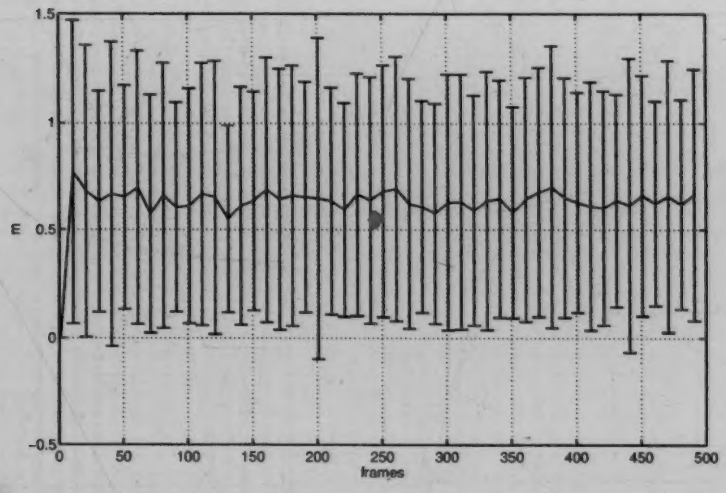


Figure 5.22: Mean and standard deviation of the absolute Z error with additional disturbances (100 simulation).

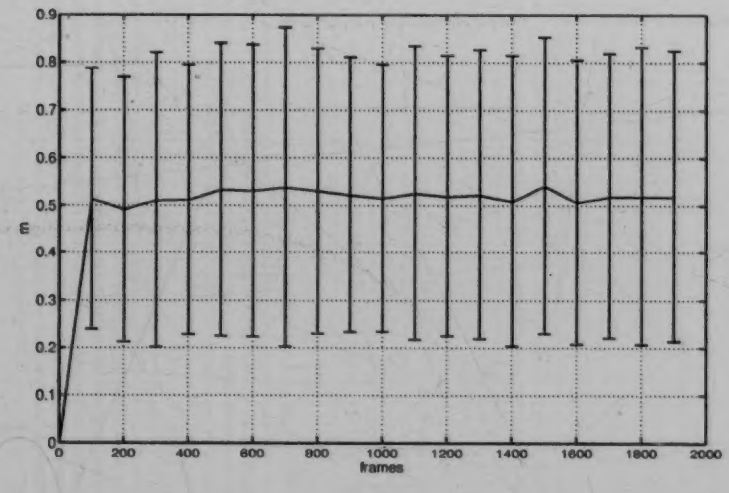


Figure 5.23: Error and standard deviation for X translations using 3D reconstruction (100 simulations).

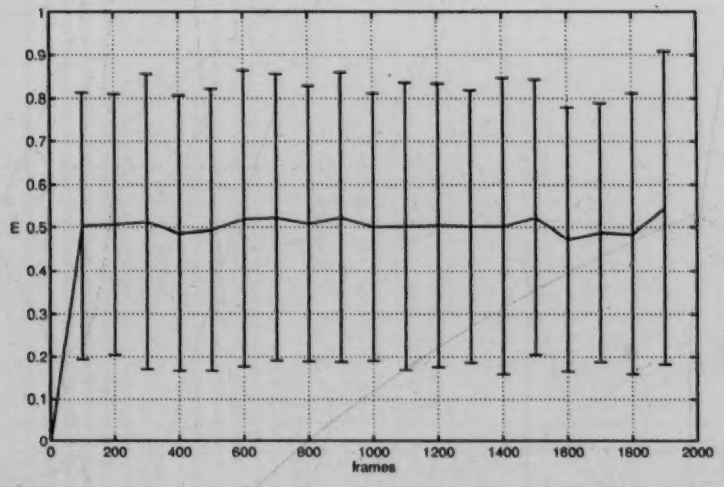


Figure 5.24: Error and standard deviation for Y translations using 3D reconstruction (100 simulations).

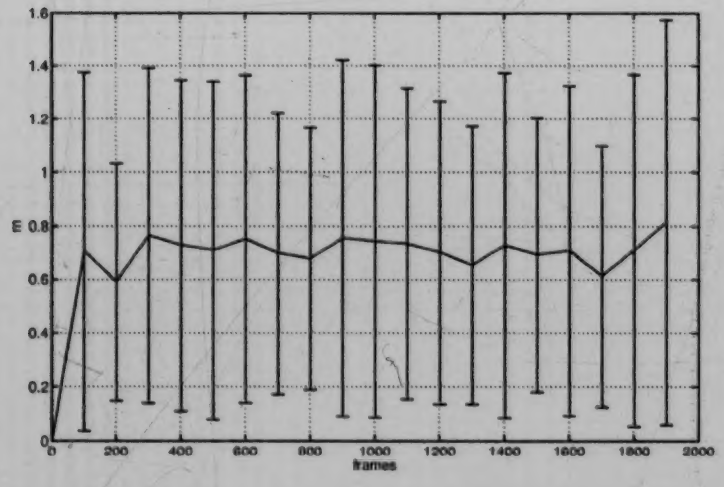
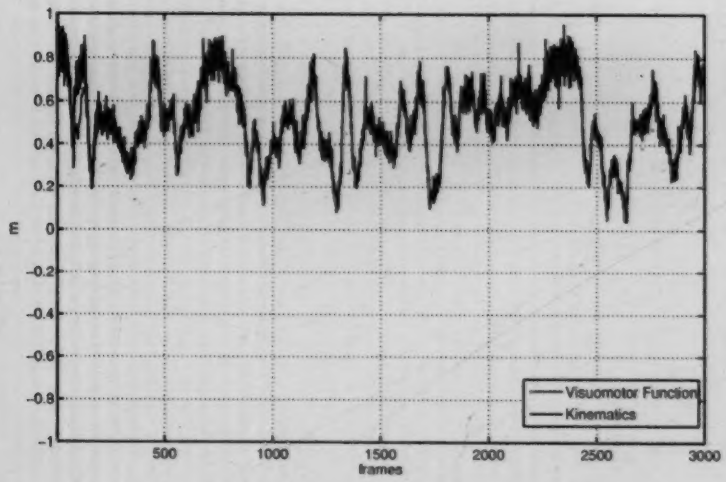
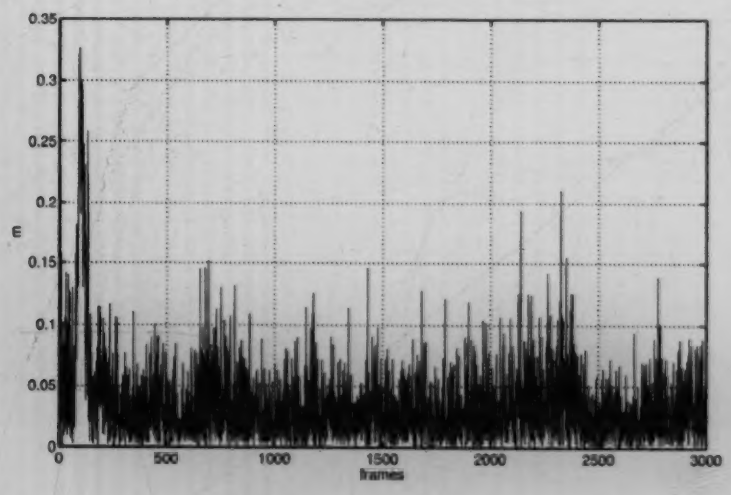


Figure 5.25: Error and standard deviation for Z translations using 3D reconstruction (100 simulations).

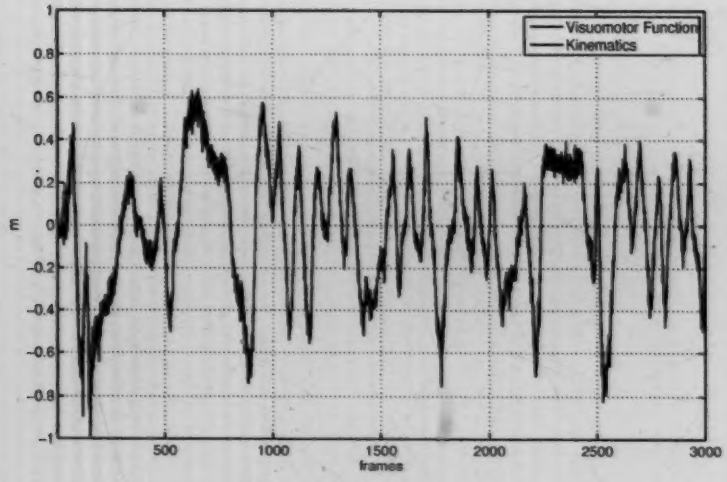


(a) Actual and estimated translation along X.

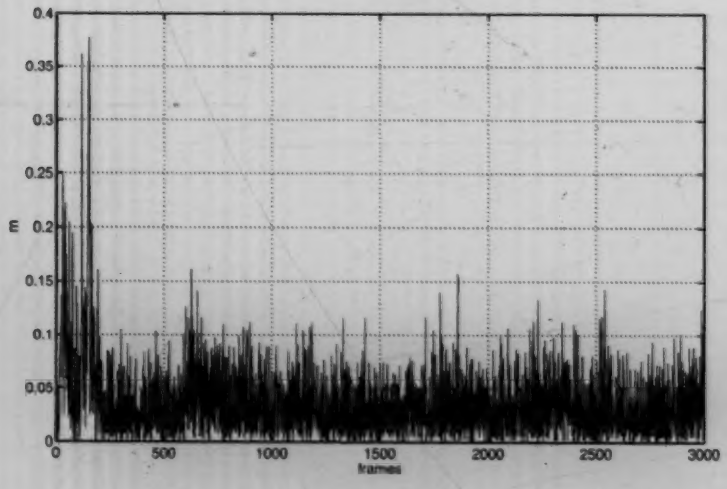


(b) Absolute translation error.

Figure 5.26: Translation along the X axis.

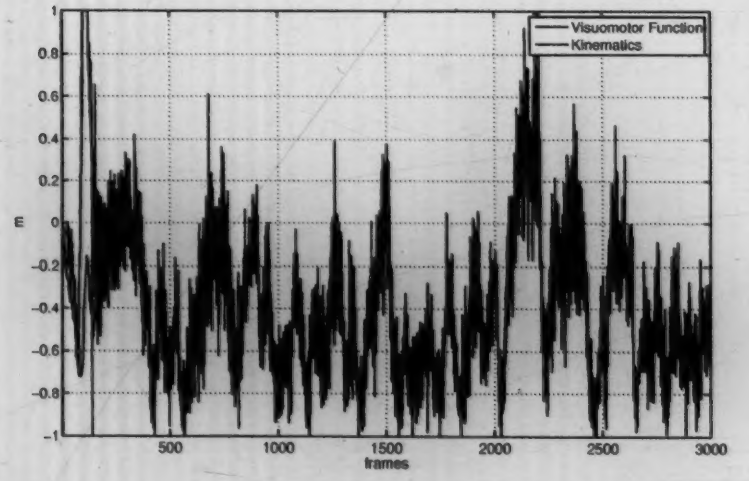


(a) Actual and estimated translation along Y.

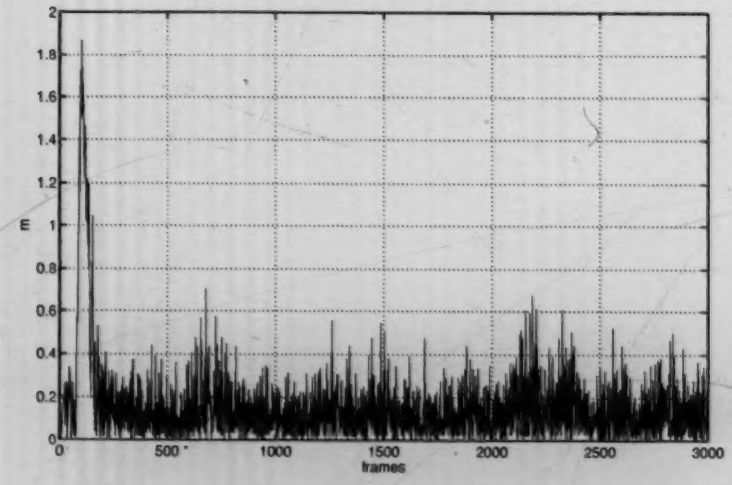


(b) Absolute translation error.

Figure 5.27: Translation along the Y axis.



(a) Actual and estimated translation along Z.



(b) Absolute translation error.

Figure 5.28: Translation along the Z axis.

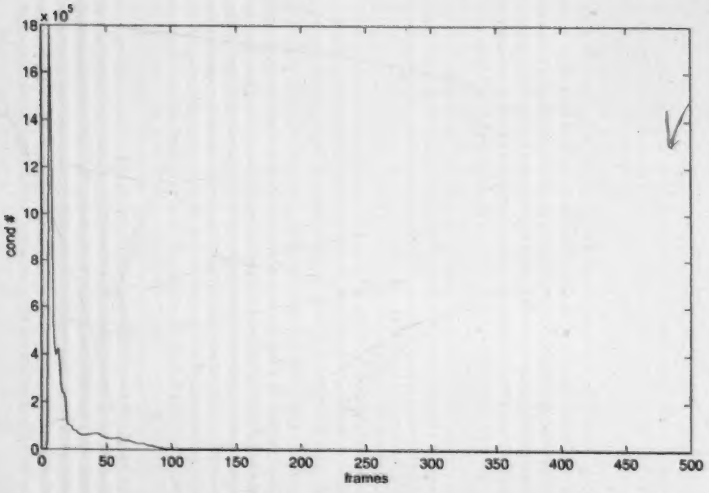


Figure 5.29: Evolution of the condition number of for one tile with real data.

5.4 6 Degrees of Freedom

Generally, control of all six degrees of freedom is the benchmark for any visual servoing method and a large amount of research in visual servoing aims at solving this problem. Several methods that address this challenge were presented in Chapter 2.

For 3D reconstruction methods, the rotation and translations can be obtained from two sets of 3D points. Again, the first step is to obtain two sets of 3D coordinates from 3D reconstruction and then to fit a special Euclidean transformation to both sets. Several methods based on least squares have been proposed to that effect [10, 182].

With the visuomotor function, the transformation between two views can be computed directly from the parameters associated with each stereo point involved. In the following experiments twelve equations where used to estimate the six degrees of freedom of a transformation.

5.4.1 Simulation: Visuomotor Function

The simulations in this section follow the procedure outlined in the previous simulation sections. For the 6 DOF case, six 3D points were randomly sampled within the 3D space delimited by $\begin{bmatrix}\pm 0.5m & \pm 0.5m & 2m \pm 0.5m\end{bmatrix}^T$. The end-effector was free to move in a $\begin{bmatrix}2m \times 2m \times 2m\end{bmatrix}$ cube and to rotate arbitrarily although only poses that projected all the points in the field of view of both cameras were selected. Each simulation involved 2,000 iterations of Algorithm 1 and a total of 100 simulations were done. Given that the 2,000 iterations results in 2,000 observations and that Algorithm 1 processes $N(N-1)$ updates based on N observations, a total of 3,998,000 calls to $UpdateParameters$ were processed during each simulation. After each iteration, the position and orientation of the end-effector with respect to the initial position was estimated by solving Equation 3.42 with the parameters $\boldsymbol{\theta}_{s_1}$ and the errors ${}^L x_i - {}^L x_1, {}^L y_i - {}^L y_1$. Figures 5.30 to 5.35 illustrate the mean absolute errors between the simulated and estimated positions of the end-effector.

All the plots show that the errors of the solution converge after 300 frames, which correspond to 600 rank-1 updates. The convergence can be verified by the condition number of the system of equations associated with a tile. For example, the plot of Figure 5.36 illustrates that the updates after the 100th frames provide little information about the visuomotor interaction. These results will now be compared to the 3D reconstruction in the next section.

One of the main challenges of using many points is that it is preferable to have every point projected in the field of view of both cameras The reason why it is preferable for a point to be visible in both cameras is to be able to index the parameters associated with the stereo point. The other reason is to measure the image-based variations that are required to update the parameters. Although failure to keep points within the field of view is not critically important as in IBVS, it defeats the

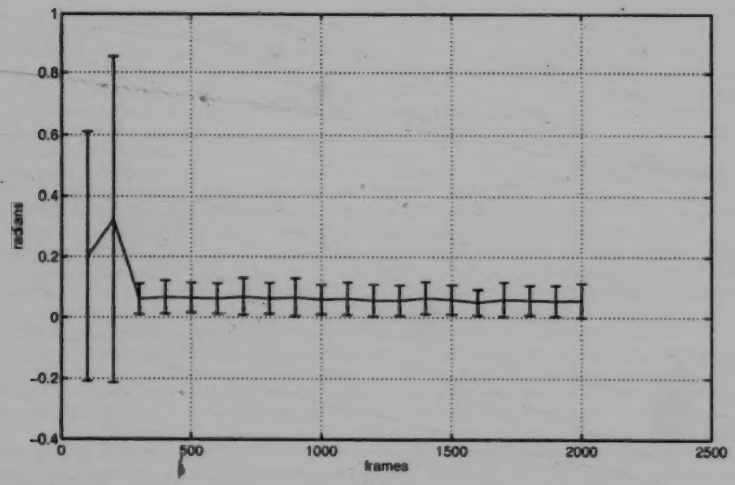


Figure 5.30: 6 DOF simulations: Error and standard deviation for rotations around the X axis (100 simulations).

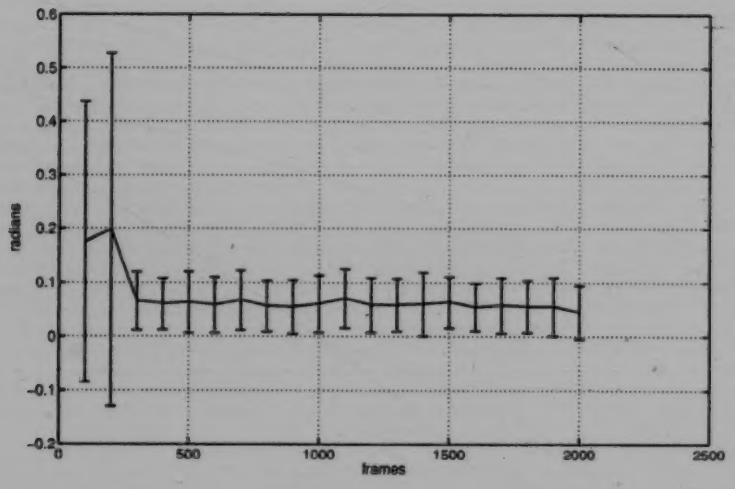


Figure 5.31: 6 DOF simulations: Error and standard deviation for rotations around the Y axis (100 simulations).

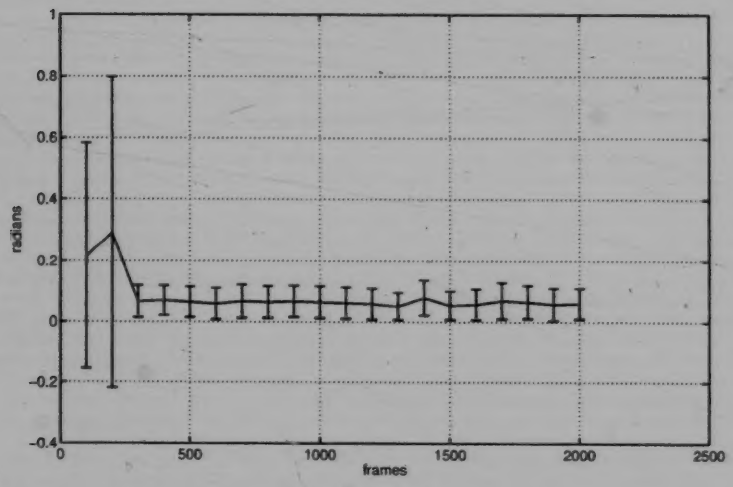


Figure 5.32: 6 DOF simulations: Error and standard deviation for rotations around the Z axis (100 simulations).

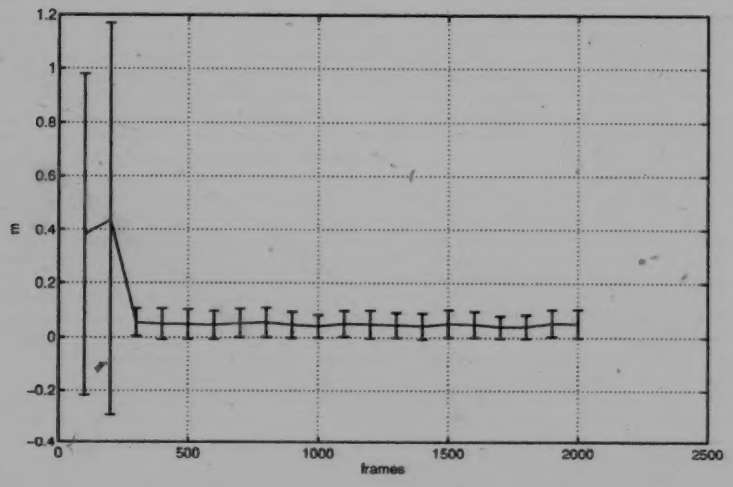


Figure 5.33: 6 DOF simulations: Error and standard deviation for X translations (100 simulations).

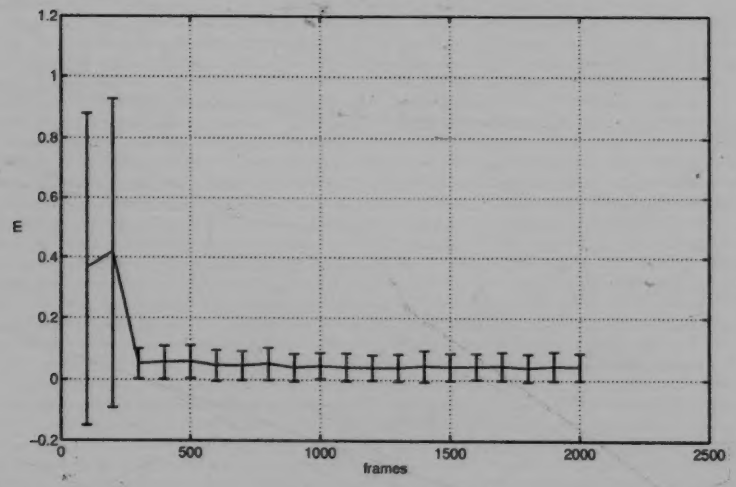


Figure 5.34: 6 DOF simulations: Error and standard deviation for Y translations (100 simulations).

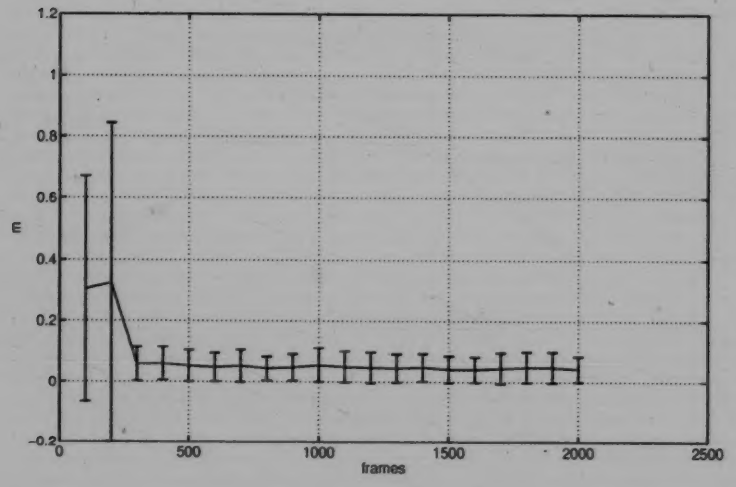


Figure 5.35: 6 DOF simulations: Error and standard deviation for Z translations (100 simulations).

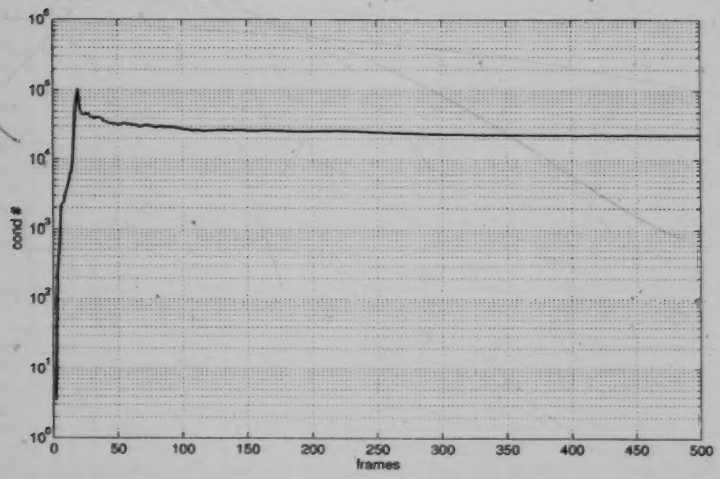


Figure 5.36: Evolution of the condition number of for one tile with simulated data.

purpose of the on line approximation algorithms. Since six points are used in the simulations, a total of 12 projections are constrained simultaneously, which inherently constrains the range of motion of the end-effector. One consequence of these constraints is that it limits the image-based variations of each target to a small area around their initial coordinates, which has an adverse effect on the distribution of updates in the stereo space.

5.4.2 Simulations: 3D Reconstruction

In this section the simulations of the visuomotor function approach for 6DOF is compared to a standard pose estimation based on 3D reconstruction. As for the translation experiments, the purpose of this experiment is to compare both approaches in simulations.

The same six 3D points used in the previous simulations were used and the same cameras and noise model were used. Again, the exact camera parameters were used to compute the 3D reconstruction. The algorithm employed for the 3D reconstruction is the same ML algorithm used in Section 5.3.

As in the previous section, the experiment consisted of one hundred simulations, each involving 2,000 transformations. The first step of each simulation is to transform the coordinates of each 3D point in the initial coordinate frame of the end-effector and then to project each point in both cameras. From these initial projections, the initial 3D coordinates of each point are estimated by the 3D reconstruction algorithm. Then, for each transformation, the coordinates of each 3D point are transformed in the new coordinate frame of the end-effector and then projected in both cameras. From these new projections, the new 3D coordinates of each point is estimated by the 3D reconstruction algorithm. Finally, the rotation and translation is estimated from the initial and new 3D coordinates. The algorithm used to estimate the rotation and translation is based on the least squares algorithm by Arun *et al.* [10].

Under Gaussian noise with a standard deviation of 3 pixels, the overall 3D reconstruction has a mean error of 0.0416m for the X coordinate, 0.0392m for the Y coordinate and 0.1135m for the Z coordinate. As expected, the Z coordinates are less accurate due to the perspective projections. Despite the relatively accurate 3D reconstruction, the least squares pose estimation between two sets of point performs poorly when compared to the visuomotor function. The results of the mean errors and standard deviations over all the simulations are illustrated for all 6 DOF in Figures 5.37 to 5.42

Compared to the corresponding graphs in Figures 5.30 to 5.35, the absolute error involved by the 3D reconstruction is roughly twice the magnitude than the error of the visuomotor approach. What is more even striking is the difference between the standard deviations which are significantly greater in 3D reconstruction. Thus, from these simulations, the accuracy and robustness of the visuomotor approach over the 3D reconstruction appears significant over the 3D reconstruction approach.

These results should be interpreted under the light that the visuomotor function bypasses the 3D

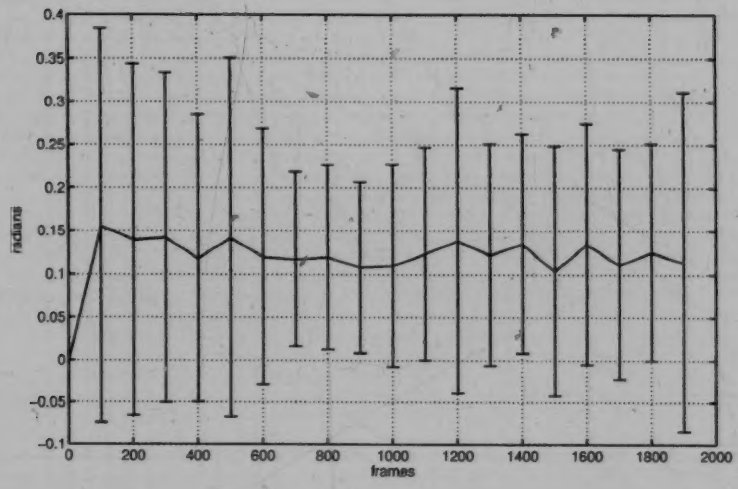


Figure 5.37: Error and standard deviation for rotations around the X axis using 3D reconstruction (100 simulations).

reconstruction whereas 3D methods require two reconstructions operations. Instead, the approximation of the visuomotor function relies on 8,000,000 equations to capture the behavior of stereo points under 6 DOF motion. From the results presented in this section, it appears that the error incurred from the reconstruction must be unrealistically small to obtain a solution comparable to the the visuomotor function.

Approximation of the Visuomotor Parameters

The CMAC algorithm generalizes the estimates of the parameters in the stereo space. That is, after each least squares update, the parameters associated with a stereo point s is generalized over a small neighborhood such that the parameters can be used to approximate the parameters of neighboring stereo points. To validate CMAC approximation, this section compares the approximation of the visuomotor parameters to the true values derived analytically.

In the first step, a simulation was executed during which 1000 3D points were sampled randomly within a $3m \times 3m$ surface in the XZ plane ($Y = 1$). Each point was projected in both cameras such that a corresponding stereo point was obtained. From the 3D coordinates and the camera parameters, the visuomotor parameters were computed analytically from Equation 3.18 and the values were propagated in the CMAC. Following the training, 100 3D test points were sampled randomly and projected in the stereo cameras. For each stereo point, the visuomotor parameters were obtained from the CMAC and compared to the analytical values. Due to the large quantity of illustrations, the results of this experiment are presented in the Appendix A.2. Results are shown only for elements that do not exibit constant values because CMAC was able to model these function exactly (errors in the range of 10^{-10}). In this experiments, the parameters that are reported are $\theta_1, \theta_2, \theta_3, \theta_7, \theta_8, \theta_9, \theta_{14}, \theta_{15}, \theta_{16}, \theta_{20}, \theta_{21}, \theta_{22}, \theta_{27}, \theta_{28}, \theta_{29}, \theta_{33}, \theta_{34}, \theta_{35}$. Results are presented in Figures A.1 to A.18. Each figure shows two illustrations. The left illustration shows the plane that results from the analytical values (i.e. the ground truth). To interpret the normal of these planes, the reader is refered to the definition of the parameters in Equation 3.18. The dots around the surface are the approximations of the parameters as given by the CMAC. In the right illustration, a histogram displays the distribution for the relative absolute error between each approximation and its true value.

5.4.3 Real

Experiments for 6 DOF motion were also conducted on the WAM. As for the PTU experiments, a hand-in-eye configuration was used (see Figure 5.9). This configuration was chosen instead of

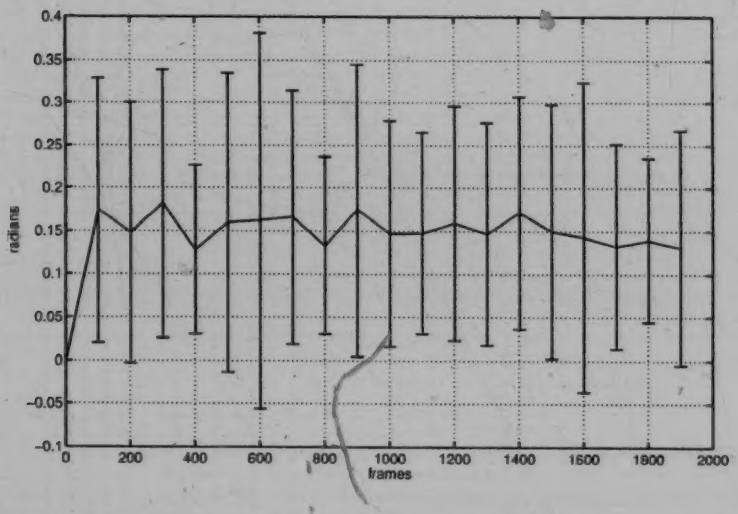


Figure 5.38: Error and standard deviation for rotations around the Y axis using 3D reconstruction (100 simulations).

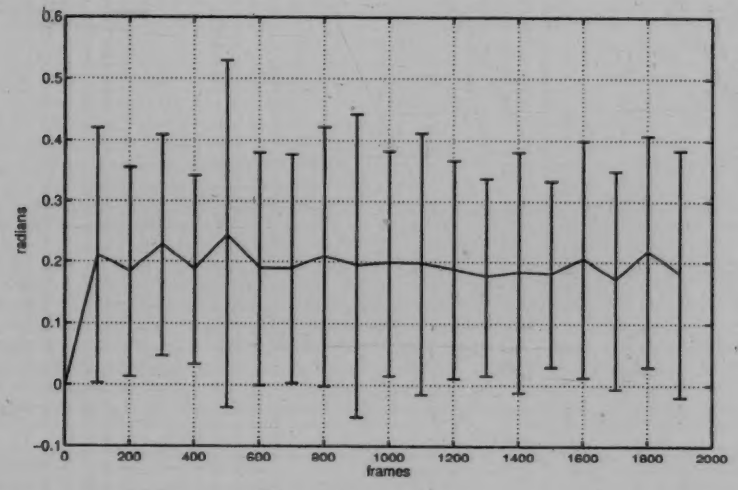


Figure 5.39: Error and standard deviation for rotations around the Z axis using 3D reconstruction (100 simulations).

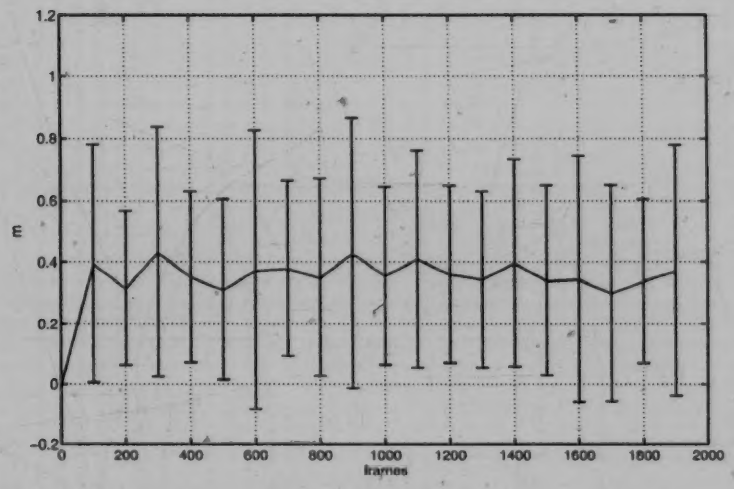


Figure 5.40: Error and standard deviation for X translations using 3D reconstruction (100 simulations).

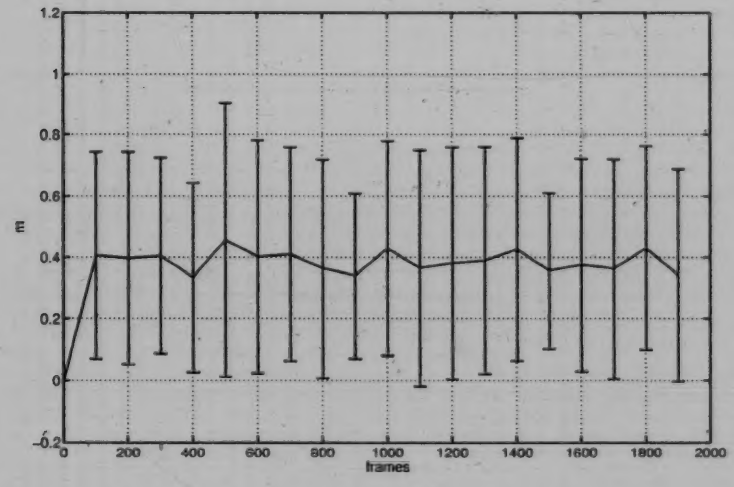


Figure 5.41: Error and standard deviation for Y translations using 3D reconstruction (100 simulations).

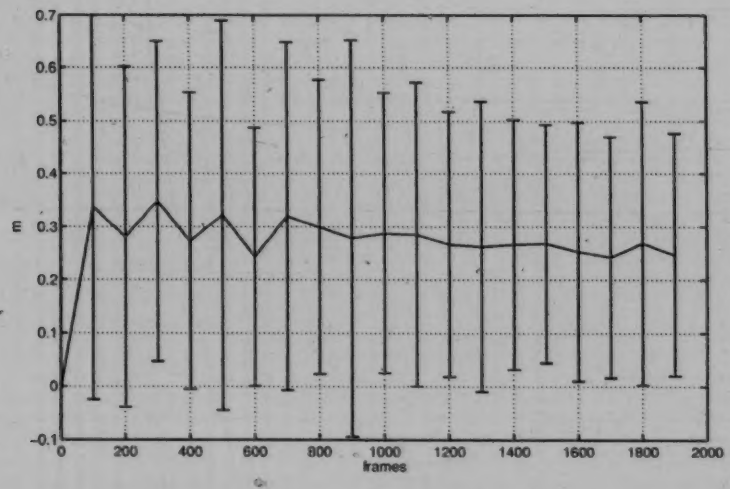


Figure 5.42: Error and standard deviation for Z translations using 3D reconstruction (100 simulations).

standalone cameras because it avoids tracking several features on the end-effector. Instead, with the eye-in-hand configuration, any feature in the environment can be tracked and this characteristic greatly simplifies the execution of the experiments.

In this experiment two visuomotor functions were used, one for each camera, and three 3D targets were involved. That is, two sets of visuomotor parameters were computed. As before, the function ${}^{L}\Theta_{s}$ was defined for the left camera. Also, the function ${}^{R}\Theta_{s}$ was defined for the right camera. It is important to note that the routine $UpdateParameters$ is the same regarless of the camera that is being used. The only difference is that the image-based errors ${}^{L}\mathbf{e}$ were used to approximate ${}^{L}\Theta_{s}$ and the image-based errors ${}^{R}\mathbf{e}$ were used to approximate ${}^{R}\Theta_{s}$. This configuration was used for this experiment because it doubles the number of equations used to solve Equation 3.42. Given that three targets are used in each camera, a total of 12 equations are used to solve Equation 3.42. If one visuomotor function would be used, then only six equations would be used.

Algorithm 1 was used one more time to process the updates. Data was collected for 3,000 frames by moving the arm manually and the pose of the end-effector was compared to the solution of Equation 3.42 after each iteration. One pipeline was assigned to each target in both cameras such that a total of six pipelines were used to process all the observations. Other than the increased number of pipelines, the procedure for this experiment was identical to the previous ones. Since 3,000 observations were collected and six pipelines were operating, the processing was done offline due to computational limitations [4]. Real-time computation for this experiment would require short pipelines and this would complicate tracking the evolution of the solution when using ${}^{L}\boldsymbol{\theta}_{*_1}$ and ${}^{R}\boldsymbol{\theta}_{*_1}$ since the initial observations would be pushed out of the pipeline after a relatively short time.

The results are presented in Figures 5.43 to 5.48. Each figure consists of two parts. The first one presents a few samples of an actual motion parameter obtained from the forward kinematics and the value obtained from the visuomotor function. The second part presents the absolute error between the actual and the estimate over the entire sequence.

Results show that the error of the solutions stabilize within approximately 400 frames. After which the solutions become relatively stable. Some of the plots show a bump around frame 700. This bump was caused by one of the trackers being blocked on the edge of an image as its target briefly exited the field of view.Also, the coordinate frames were oriented differently due to the hand-eye configuration. As such, the X axis of the end-effector represents the Z axis in the camera frame and the Z axis of the end-effector represent the Y axis in the camera frame. The mean and standard

[4]When a single pipeline is used, a rough maximum of 2000 updates can be processed per core per second. In the experiments, one core was used for each camera such that three pipelines were operating on each core.

Table 5.2: Mean absolute error and standard deviation for 6 DOF with real data.

	R_x (rad)	R_y (rad)	R_z (rad)	t_x (m)	t_y (m)	t_z (m)
mean absolute error	0.0646	0.0450	0.0521	0.0866	0.0921	0.0760
standard deviation	0.0449	0.0355	0.0466	0.0669	0.0655	0.0621

deviation after the 800th frame (after the bump) are shown in Table 5.2.

As for most experiments with real data, the density of the updates was concentrated around the center of the images (Figures 5.49). Again, this reflects the difficulty of distributing the updates when manipulating the arm manually. On the plus side, this allowed to collect a large amount of data for a relatively small area, which might be factor in the results.

5.5 Mobile Robot

Mobile robots are widely used in research and they are making inways in households and offices at a much faster pace than their manipulators counterparts. Yet, mobile robots, and in particular non-holonomic ones pose a singular problem for IBVS methods. Because typical non-holonomic mobile robots cannot move laterally, IBVS cannot regulate tasks that involve only lateral movements. Furthermore, the motion of non-holonomic robots often involve complex paths or trajectories that do not consider the visibility of each target.

A large amount of research involving mobile robots aims at building maps from stereo images [63]. These maps are used by the robot to navigate and avoid obstacles. Typically, the maps are composed of 3D coordinates (often 2D) corresponding to visual landmarks. Among the available landmarks, Scale-invariant feature transform (SIFTs) [123] have been the most successful used in mapping applications.

The experiments were executed on a RWI Magellan robot. Two cameras were mounted on the front of the robot roughly 20cm apart. First, the solution to Equation 3.45 is compared to measurements obtained from the odometry. Finally, an example of a visual servoing task is presented.

Evaluation in Cartesian Space

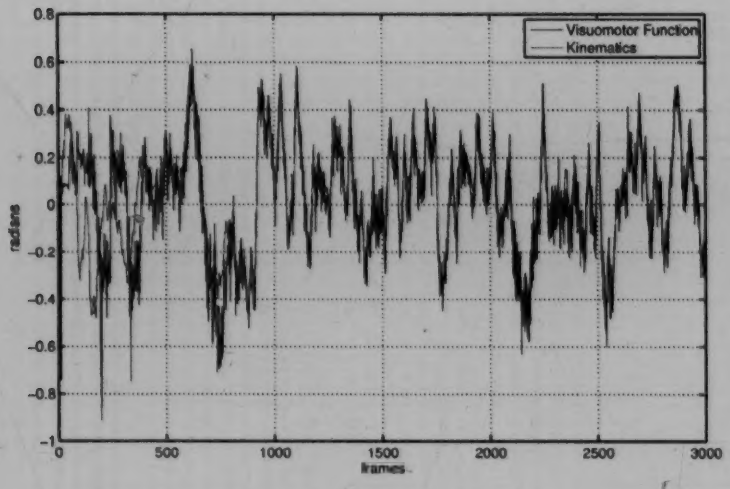
The first experiment tests the solutions of the visuomotor function for positioning tasks. Algorithm 1 was used for 512 iterations During this period of time the robot was move manually and two targets were tracked simultaneously in both cameras.

Then, the solution to Equation 3.45 was used to estimate the relative displacement of the robot with respect to the initial position and the solution was compared to the odometry measurements. initial state. Results for the elements r_{11}, r_{21}, t_x and t_y are presented in Fig. 5.50.

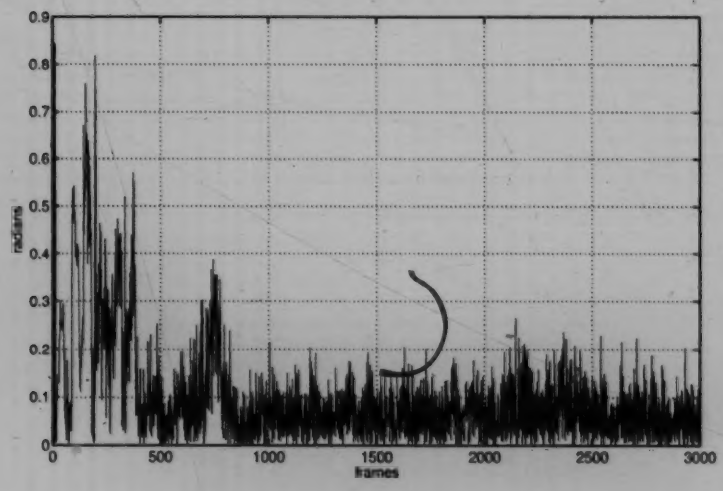
Visual Servoing with Feed-forward

The last experiment demonstrates an example of a mobile robot using the solution of the visuomotor function to do feed-forward motion control. For this, the system relies on SIFT [123] to match features between frames. The diagram of the system is illustrated in Figure 5.51. The first stage consists of extracting SIFTs in both cameras. Because the visuomotor function requires stereo points, the features in each images were matched to obtain stereo correspondence. Since the cameras were installed manually, the search for stereo correspondence was exhaustive because the search could not be constrained to the usual horizontal epipolar lines. Given the stereo matches, the second stage obtains observations to update the visuomotor function. To obtain image-based variations and displacements of the robot, each SIFT is matched against a database of keypoints. Each entry in a database includes the following

1. SIFT descriptor for matching
2. Most recent position or the robot from which the feature was observed
3. Most recent image coordinates the feature was observed
4. An odometry stamp.

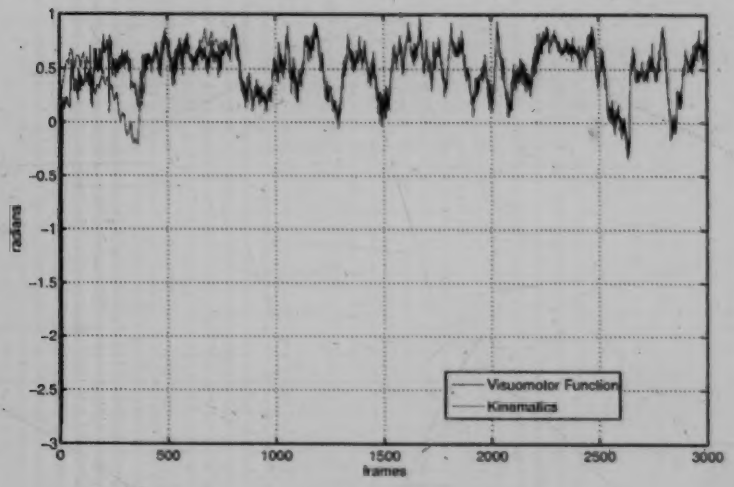


(a) Rotations around X.

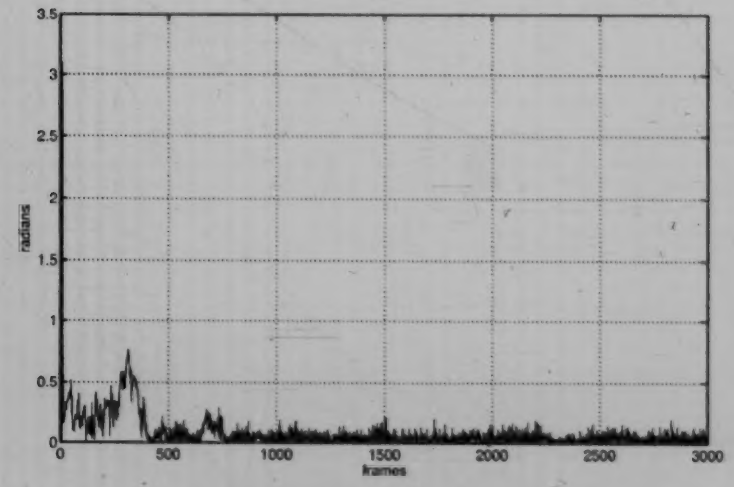


(b) Absolute error of rotations around X.

Figure 5.43: 6DOF: Rotations around X.

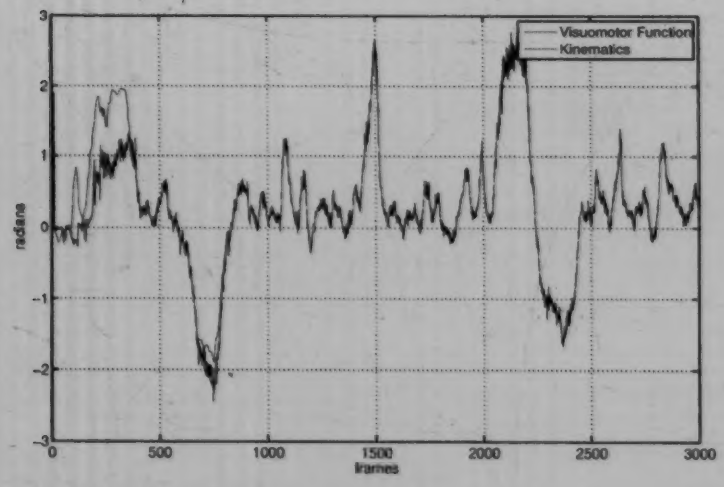


(a) Rotations around Y.

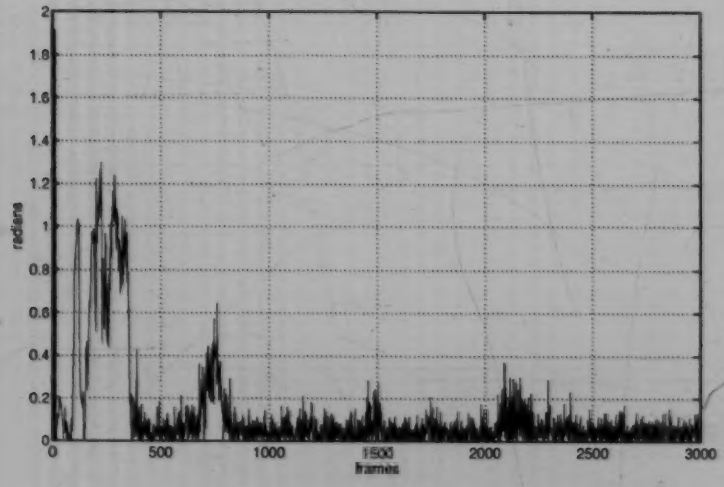


(b) Absolute error of rotations around Y.

Figure 5.44: 6DOF: Rotations around Y.

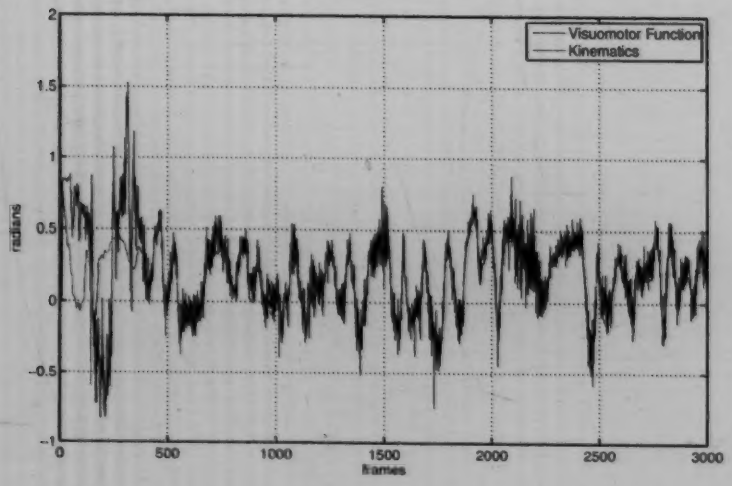


(a) Rotations around Z.

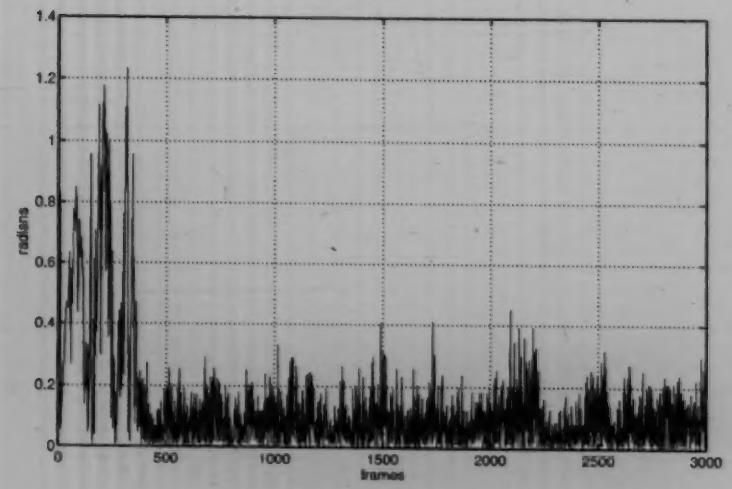


(b) Absolute error of rotations around Z.

Figure 5.45: 6DOF: Rotations around Z.

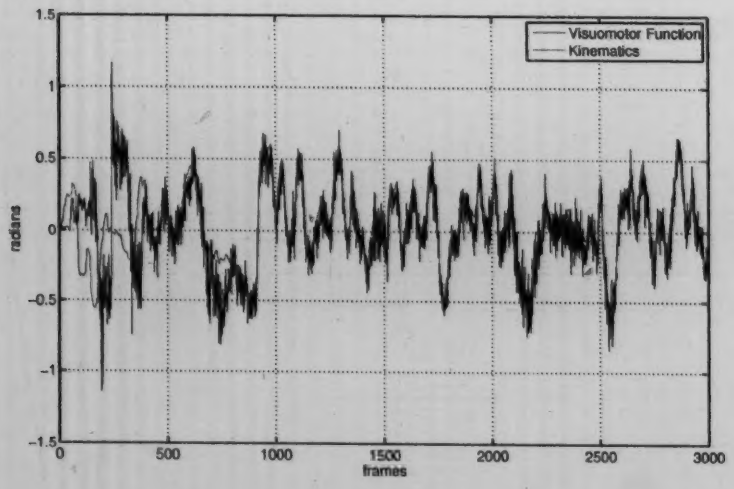


(a) X translations.

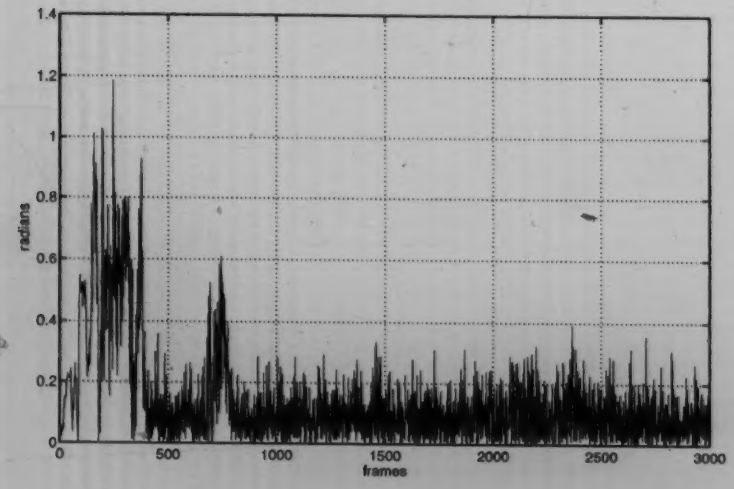


(b) Absolute error of X translations.

Figure 5.46: 6DOF: Translations along the X axis.

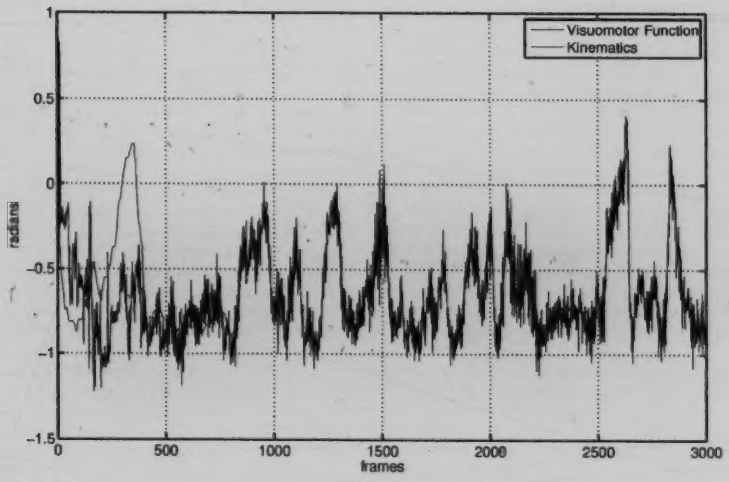


(a) Y translations.

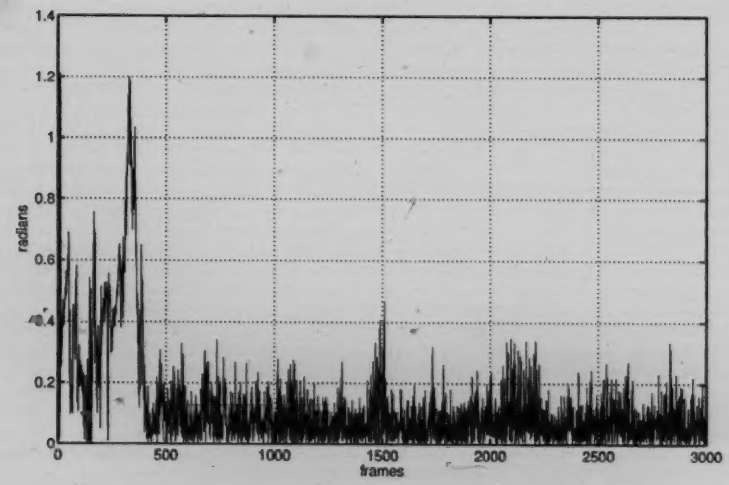


(b) Absolute error of Y translations.

Figure 5.47: 6DOF: Translations along the Y axis.

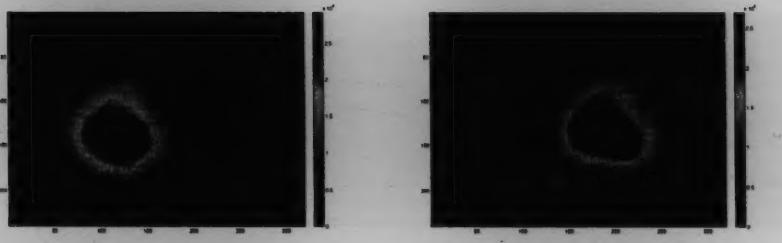


(a) Z translations.



(b) Absolute error of Z translations.

Figure 5.48: 6DOF: Translations along the Z axis.



(a) Update map for the left camera from real data. (b) Update map for the right camera from real data.

Figure 5.49: Update map for 6DOF. The intensity of a pixel reflects the amount of updates.

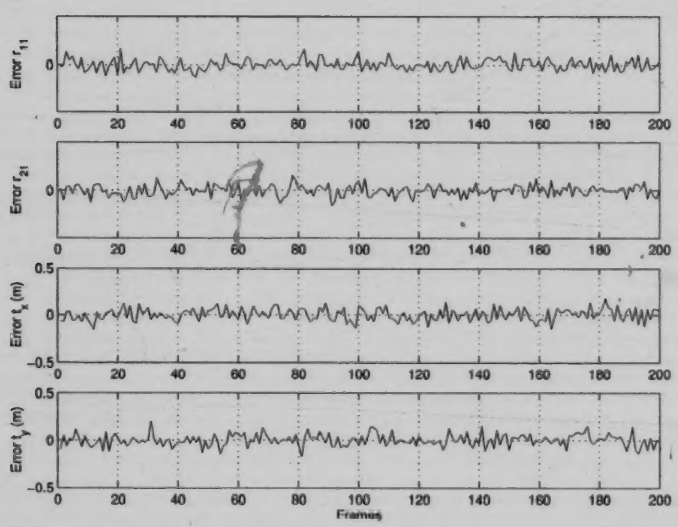


Figure 5.50: Error between the motion estimation of the visuomotor function and the odometry.

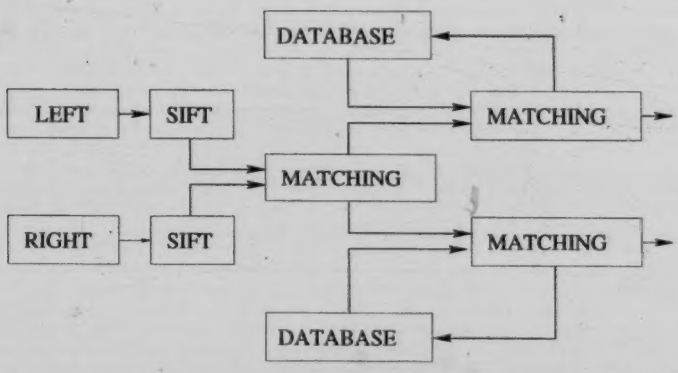


Figure 5.51: Flow chart for processing and matching SIFT keypoints.

The odometry stamp was used as an expiration date on the feature to avoid matches with old features due to the drifting of the odometry. Basically, any match that involved a feature that was detected more than 3m away was voided. This was necessary to avoid corrupting the parameters with inaccurate data. The time stamp was determined empirically during unrelated experiments where it was determined that the robot could go on a reasonable straight line for roughly 3m. Thus, because of the time stamps, Algorithm 1 was not used and updates were performed only between the matched features.

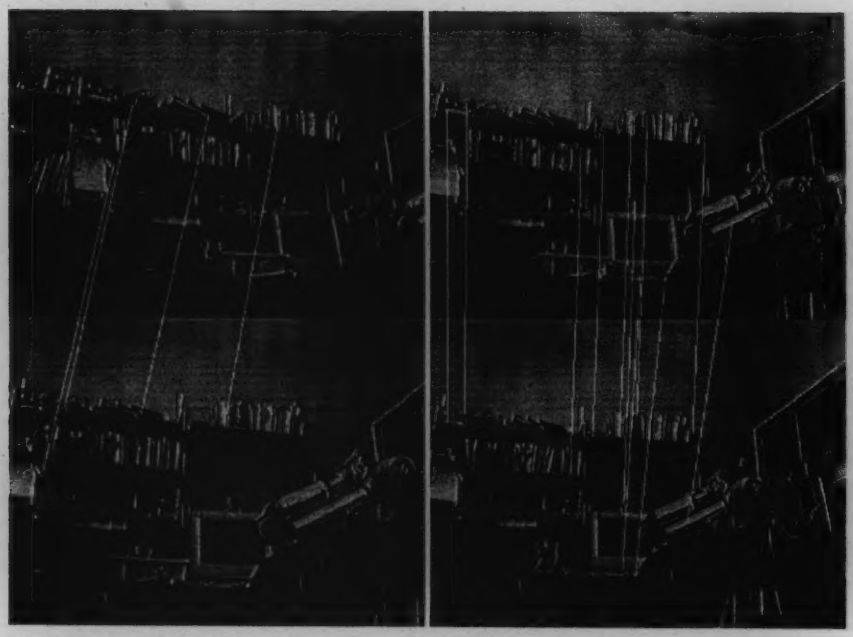
A feature that did not find a match in the database was added to the database only if its descriptor was sufficiently distinctive from the others. If a SIFT did find a match in the database, then the image and position variations where used to update the parameters associated with the stereo points.

The robot was tethered to a dual core 3GHz desktop. Each camera used a separate core to extract and match the SIFT at around 5 frames per second. The SIFT parameters were tuned to produce around 200 keypoints per frame and between 10 to 50 stereo matches. Matches from the database typically varied between 0 and 15.

The training stage consisted of moving the robot manually for an extended period of time within an area of approximately $12m^2$ in a room. During that time about 1000 keypoints were detected in the environment and added to the databases. These keypoints were used to establish correspondences between frames. When a match was successfull in both cameras, then the image-based error was determined and the visuomotor parameters of the keypoint were updated. After the training, the robot was moved to its desired position and a snapshot was taken from which SIFT keypoints were extracted (bottom of Figure 5.52(a)). Finally, the robot was moved to its starting position. Then, another snapshot was taken and the keypoints were extracted (top of Figure 5.52(a)). The matching between initial and final keypoints resulted in 4 correspondences. Using these matches, the system estimated the transformation between the two views by solving Equation 3.45. This transformation was supplied to the position controller implemented for this experiments[168]. The target destination was marked on the floor at $(-0.5m, -1.5m)$ away from the initial position with 0 heading (a pure translation). The motion of the robot is shown in Figure 5.53 and the approximate destination position is marked with a red square. During the motion, the robot completely lost visual contact with the destination, but, as it moved closer to the goal, more features were matched between the current and desired views.

To avoid instability of the position controller, the motion was stopped when the odometry reached a 2.5cm radius around the destination and within 0.025 radian of the desired orientation. After the motion was stopped, a final snapshot was taken and the images were matched against the command images. This resulted in 11 matches and the average absolute error of their coordinates were 15 pixels for the x coordinates and 7 for the y coordinates. However, the robot stopped its trajectory with an error of 0.23 radian and 0.17m along the X axis and $0.21m$ along the Y axis.

One of the main problem in this experiment is the combination of the non-linear drift of the odometry. Using a technique often used in SLAM [63], the robot was move in straight lines and then was fully rotated to increase the number of matches. Nevetheless, it was extremely difficult to obtain consistent odometry readings when moving over a distance greater than 3m. Therefore, every motion segment were kept short.



(a) Matches between the initial image and the command image.

(b) Matches between the final image and the command image.

Figure 5.52: SIFT matches.

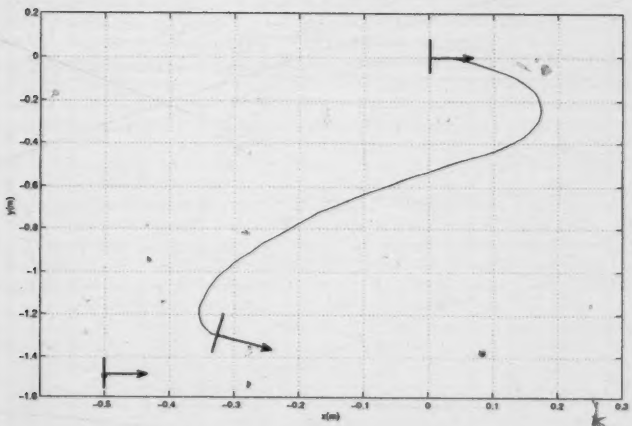


Figure 5.53: Trajectory of the robot.

Chapter 6

Conclusion

After several decades of research, visual servoing has become an established area within robotics. The strength of visual servoing methods is that their stability can often be established through Lyapunov stability theorem. Although visual servoing has demonstrated its use in research environments, its deployment in commercial or industrial products has been sporadic. One reason for this slow transfer is the incompatibility between the output of visual servoing control laws and the input of commercial manipulators and path planning algorithm. Whereas the majority of visual servoing control laws span the space of twists, the input of most path planners and commercial robots span the space of special Euclidean transformations. One argument for using $SE(3)$ commands is that it makes path planning and trajectories deterministic. That is, the motion of the arm can be fully determined ahead of its execution. A similar argument holds for autonomous robots. Undoubtedly, these robots will be required to use their sensors, including vision, to guide their motion and do tasks in natural environments. For many reasons, safety being one of them, these robots will be required to plan tasks ahead of executing them.

To plan a task in $SE(3)$, a goal position must be determined in $SE(3)$. Furthermore, the command should not only account for the sensor that is used, but also for the interaction between the actuator, the sensor and the environment. The coupling actuator/sensor has been the cornerstone of most of the research in visual servoing where the visual feedback is directly coupled to the actuating. In feed-forward systems, however, the trend is to use the 3D space as the common denominator between the sensor and the actuator. That is, feed-forward systems often use the 3D reconstruction as the means rather than the objective. As such, much of the emphasis in feed-forward system has been on interpreting the 3D space from images.

Recognizing the gap between IBVS and look-then-move PBVS, this thesis proposes an image-based feed-forward system called the visuomotor function. From the perspective of visual servoing, the visuomotor function is an interaction matrix that maps elements of $SE(3)$ to image-based variations. As such, the visuomotor function provides the feed-forward domain of of look-then-move systems and preserves the image-based range of IBVS control laws.

The theory around the visuomotor function involves the definition of the visuomotor camera and its interaction with 3D points. Compared to passive cameras, the visuomotor camera is an active camera that projects the variations of 3D points caused by the displacement of the camera. Because of this property, the visuomotor function provides a set of visuomotor parameters for each 3D point. These parameters capture the behavior of each point for the rigid displacement of the camera. That is, given the visuomotor parameters of a 3D point and an image-based variation, the displacement of the end-effector is determined by the solution of the visuomotor function. The visuomotor function of four tasks often used in robotics were presented. Namely they are: six degrees of freedom motion, pan and tilt, 3D translations and planar motion. Although these function have the same structure, the dimensionality of the parameters space is reflected by the complexity of the interaction between the camera and the environment. The visuomotor function can also be used for feedback system at the cost of trading the $SE(3)$ for velocities as the domain of the control law. This thesis showed the asymptotic stability of the visuomotor function.

These visuomotor parameters are estimated on-line through a stable incremental least squares based on QR factorization. This algorithm enables the system to approximate the parameters of the visuomotor system on-line. The approximation is executed in real-time as the data is made available.

Since each 3D point is associated to a set of visuomotor parameters a method for indexing these parameters from images coordinates is presented. Essentially, the proposed solution relies on the stereo coordinates of the 3D point. Compared to 3D reconstruction, however, the stereo cameras are not used to estimate 3D coordinates but to index the parameters of each point in a database. Because the parameters are continuously updated they are tuned to capture the image-based reaction of 3D

volumes to the motion of the camera. This reflects the distinction between 3D reconstruction and the visuomotor function approach. Whereas the former aims at representing the visual input by 3D coordinates, the later is only concerned by the interaction between the camera and the environment.

Nevertheless, the domain of stereo coordinates is large and storing a distinct set of visuomotor parameters for every stereo point poses several challenges. To address these challenges, the stereo space is approximated by a CMAC algorithm. The benefits of the CMAC are two fold. First it avoids storing a large set of parameters. Second, and most importantly, it allows to generalize the parameters associated to a stereo point to neighboring stereo points. This second item avoids the necessity to estimate the visuomotor parameters associated with every stereo point.

To cope with the amount of visuomotor parameters to estimate, an algorithm was presented to increase the amount of updates between observations. The algorithm is based on a pipeline that inserts the observations in a list as they become available. Then, the algorithm updates each set of parameters represented in the list from the new observation and, conversely, the new set of parameters represented by the new observation is updated from each entry in the list. The number of updates per frame grows linearly with the number of entries in the list.

Experiments demonstrated the performance of the visuomotor approach for the different tasks presented in this work. The experiments involved hundreds of simulations and real experiments that involved the computation of thousands of solutions. In the case where a direct comparison was possible, the visuomotor function did significantly better than motion estimation based on 3D reconstruction. The reason for these performances are two fold. First, the visuomotor function captures the interaction of the cameras with its environment. Since feed-forward control involves the motion of the camera in $SE(3)$, the visuomotor function is able to capture the nature of this interaction. Second, the displacement of the camera is given directly from the solution to the visuomotor function. Thus, the solution is the result of only one optimization. The optimization algorithm depends on the nature of the problem and the Levenberg Marquardt algorithm was used for non-linear least squares problems and linear least squares was used otherwise. Compared to this, solutions obtained from 3D reconstructions required three optimizations. The first one is the estimation of the initial 3D coordinates of each point. The second is the estimation of the final 3D coordinates of each point. The last one is the estimation of the 3D motion given the 3D estimates.

Based on these results, the visuomotor function is a good alternative for feed-forward control and complements visual servoing by using the same error vector than IBVS methods. Interestingly, what would normally appear as the Achilles' heel of the visuomotor function is in fact its strength. Despite the large number of parameters and the generally negative perception of camera calibration, the parameters enable the visuomotor function to approximate very specific behaviors. Combined with the pipeline algorithm, the error of the solutions converges within a few seconds. In fact, if all the observations are queued, then the number of rank-1 updates processed after t frames is $2t^2$. Thus, at 30fps the algorithms generates 6,480,000 rank-1 updates for one minute of training.

Among the possible improvements to the visuomotor function is the relative sensitivity to noise. The main source of this sensitivity was identified as the presence of e_x and e_y in the right hand side of Equations 3.13, 3.21, 3.28 and 3.35. Although the amount of noise involved in the experiments was reasonable, simulations have shown that the algorithm performs extremely well when the noise is isolated to the right-hand side error vector $\mathbf{e}$. Even disturbing by a small amount the e_x and e_y that enter in the matrices typically produced better results. A possible algorithm to overcome this problem is partial total least squares [99]. The algorithm, however, requires an SVD factorization and incremental SVD algorithms are more involved than the ILS presented in Section 3.3.

Another source of possible improvement would be a CMAC with a multi-resolution grids. One observation in the approximation of the visuomotor parameters is that the convergence of the condition number is a good indicator for the convergence of the parameters. Thus, the system could use the parameters of tiles with higher resolutions when these parameters have converged or use tiles with a lower resolution when tiles at higher resolutions have yet to be updated.

A more involved venture would be to move away from the formalism of $SE(3)$ and define the visuomotor function as the relation between image-based variations and variations in joint space. This would avoid any dependence on kinematics parameters and calibration. For example, each time that a robot changes its tool control point, the visuomotor function must be re-approximated because of the change in the kinematics chain.

Finally, it is important to mention that whether it is feed-forward or feedback, vision-guided robotics is only as good as the hand-eye configuration allows it to be. It should be remembered that behind most image-based command there is an actual manipulation: picking an object, pressing a button, etc. Often, these manipulations will fail even though the visual task has been accomplished. Two explanations for this are the poor conditioning of image-based tasks and the resolution of the cameras. One approach to address these challenges is to control the viewpoint of the camera position the camera in order to optimize the success of the manipulation. This optimal viewpoint control is not considered as visual servoing *per se*, but rather a method that improves the conditioning of a visual servoing task. Humans are very proficient at this sort of process as we naturally do

fine manipulations by monitoring the tasks from adequate viewpoints. Yet little research has been invested in a solution to this problem, even though it could lift a burden of on the shoulder of visual servoing.

Bibliography

[1] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(4):384–401, 1985.

[2] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Journal of Dynamic System, Measurement, and Control*, pages 220–227, September 1975.

[3] Ali Alhaj, Christophe Collewet, and François Chaumette. Visual servoing based on dynamic vision. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, volume 3, pages 3055–3060, September 2003.

[4] Peter K. Allen, Aleksandar Timcenko, Billibon Yoshimi, and Paul Michelman. Automated tracking and grasping of a moving object with a robotic hand-eye system. *IEEE Transactions on Robotics and Automation*, 9(2):152–165, April 1993.

[5] Benedetto Allotta and Carlo Colombo. On the use of linear camera-object interaction models in visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):350–357, April 1999.

[6] Nicolas Andreff, Bernard Espiau, and Radu Horaud. Visual servoing from lines. *The International Journal of Robotics Research*, 21(8):679–699, August 2002.

[7] Nicolas Andreff, Radu Horaud, and Bernard Espiau. Robot hand-eye calibration using structure-from-motion. *The International Journal of Robotics Research*, 20(3):228–248, March 2001.

[8] Jorge Angeles, Gilbert Soucy, and Frank P. Ferrie. The online solution of the hand-eye problem. *IEEE Transactions on Robotics and Automation*, 16(6):720–731, December 2000.

[9] S. Arimoto, S. Kawamura, and F. Miyazaki. Bettering operations of robots by learning. *Journal of Robotic Systems*, 1(2):123–140, 1984.

[10] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 9(5):698–700, September 1987.

[11] M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 1996.

[12] Koichi Ashimoto. A review on vision-based control of robot manipulators. *Advanced Robotics*, 17(10):969–991, 2003.

[13] Robotic Industries Association. Robotics industry sets new records in 2005 as new orders jump 23% in north america. http://www.roboticsonline.com/public/articles/articlesdetails.cfm?id=2283, August 2006.

[14] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[15] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning for control. *Artificial Intelligence Review*, 11:75–113, 1997.

[16] Abílio Azenha. Iterative learning in variable structure position/force hybrid control of manipulators. *Robotica*, 18:213–217, 2000.

[17] John S. Baily. Adaptation to prisms: Do proprioceptive changes mediate adapted behaviour with ballistic arm movements? *The Quarterly Journal of Experimental Psychology*, 24(1):8–20, 1972.

[18] C. Baroglio, A. Giordana, M. Kaiser, M. Nuttin, and R. Piola. Learning controllers for industrial robots. *Machine Learning*, 1996.

[19] Peter A. Beckett. Development of the third component in prism adaptation: Effects of active and passive movement. *Journal of Experimental Psychology*, 6(3):433–444, August 1980.

[20] Jeffrey S. Beis and David G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Puerto Rico, June 1997.

[21] Margrit Betke and Leonid Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, April 1997.

[22] Daniel L. Boley, Erik S. Steinmetz, and Karen T. Sutherland. Robot localization from landmarks using recursive total least squares. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 1381–1386, Minneapolis, April 1996.

[23] Robert Bolles and Richard Paul. The use of sensory feedback in a programmable assembly system. Technical Report CS-396, Stanford, October 1973.

[24] Michael Brady, John M. Hollerbach, Timothy L. Johnson, Tomas Lozano-Perez, and Matthew T. Mason, editors. *Robot Motion: Planning and Control*. The MIT Press, 1982.

[25] R.A. Brooks. Intelligence without representation. *Artificial Intelligence Journal*, 47:139–159, 1991.

[26] M. Brown and C.J. Harris. Least mean square learning in associative memory networks. In *Proceedings of the 1992 International Symposium on Intelligent Control*, pages 531–536, Glasgow, UK, August 1992.

[27] M. Brown and C.J. Harris. The modelling abilities of the binary cmac. In *Proceedings of the 1994 International Conference on Neural Networks*, volume 3, pages 1335–1339, Orlando, June 1994.

[28] Martin Brown, Christopher J. Harris, and Patrick C. Parks. The interpolation capabilities of the binary cmac. *Neural Networks*, 6(3):429–440, 1993.

[29] Martin Brown and C.J. Harris. Comments on "learning convergence in the cerebellar model articulation controller". *IEEE Transactions on Neural Networks*, 6(5):1016–1018, July 1995.

[30] E. Burdet and J. Luthiger. Coordination learning of robot movements with vision processes. *Robotica*, 1999.

[31] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proceedings of the 1996 IEEE/RSJ Internatinoal Conference on Intelligent Robots and Systems*, 1996.

[32] François Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G. Hager, and A.Morse, editors, *The Confluence of Vision and Control*, volume 237 of *Lecture Notes in Control and Information Systems*, pages 66–78. Springer-Verlag, 1998.

[33] François Chaumette. Image moments: a general and useful set of features for visual servoing. *IEEE Transactions on Robotics*, 20(4):713–723, August 2004.

[34] François Chaumette, Samia Boukir, Patrick Bouthemy, and Didier Juvin. Structure from controlled motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):492–504, May 1996.

[35] François Chaumette and Seth Hutchinson. Visual servo control part i: Basic approaches. *IEEE Robotics & Automation Magazine*, 13(4):82–90, December 2006.

[36] François Chaumette and Seth Hutchinson. Visual servo control part ii: Advanced approaches. *IEEE Robotics & Automation Magazine*, 14(1):109–118, March 2007.

[37] Yiu Cheung Shiu and Shaheen Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transorm equations of the form $ax = xb$. *IEEE Transactions on Robotics and Automation*, 5(1):16–29, February 1989.

[38] Chong Sook Choe and Robert B. Welch. Variables affecting the intermanual transfer and decay of prism adaptation. *Journal of Experimental Psychology*, 102(6):1076–1084, June 1974.

[39] J.C. Clarke and A. Zisserman. Detection and tracking of independant motion. *Image and Vision Computing*, 14:565–572, 1996.

[40] Dottie M. Clower, John M. Hoffman, John R. Votaw, Tracy L. Faber, and Roger P. Woods. Role of posterior parietal cortex in the recalibration of visually guided reaching. *Nature*, 383:618–621, October 1996.

[41] C. Colombo, B. Allotta, and P. Dario. Affine visual servoing: A framework for relative positioning with a robot. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 1, pages 464–471, May 1995.

[42] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–575, November 2003.

[43] Andrew I. Comport, Eric Marchand, Muriel Pressigout, and François Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, July/August 2006.

[44] Fabio Conticelli and Benedetto Allotta. Nonlinear controllability and stability analysis of adaptive image-based systems. *IEEE Transactions on Robotics and Automation*, 17(2):208–214, April 2001.

[45] Peter I. Corke and Malcolm C. Good. Dynamic effects in visual closed-loop systems. *IEEE Transactions on Robotics and Automation*, 12(5):671–683, October 1996.

[46] Peter I. Corke and Seth A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, August 2001.

[47] P.I. Corke. The unimation puma servo system. Technical Report MTM-226, CSIRO, July 1994.

[48] A. Crétual and F. Chaumette. Application of motion-based visual servoing to target tracking. *The International Journal of Robotics Research*, 20(11):878–890, 2001.

[49] Armel Crétual and François Chaumette. Visual servoing based on image motion. *The International Journal of Robotics Research*, 20(11):857–877, 2001.

[50] Daniel Crevier. *AI: The Tumultuous History of the Search for Artificial Intelligence*. BasicBooks, 1993.

[51] Andrew J. Davison, Ian D. Reid, Nicholas D. Moton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1–16, June 2007.

[52] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry Algorithms and Applications*. Springer, 1997.

[53] Koichiro Deguchi. Optimal motion control for image-based visual servoing by decoupling translation and rotation. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 705–711, October 1998.

[54] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, volume 2, pages 1322–1328, May 1999.

[55] D. DeMenthon and L.S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15:123–141, 1995.

[56] A.P Dempster, N. M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society. Series B*, 39(1):1–38, 1977.

[57] Lingfeng Deng, Farrokh Janabi-Sharifi, and William J. Wilson. Hybrid motion control and planning strategies for visual servoing. *IEEE Transactions on Industrial Electronics*, 52(4):1024–1040, August 2005.

[58] C. Distante, A. Anglani, and F. Taurisano. Target reaching by using visual information and q-learning controllers. *Autonomous Robots*, 2000.

[59] Fadi Dornaika and Radu Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Transactions on Robotics and Automation*, 14(4):617–622, August 1998.

[60] Arnaud Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report TR.310, University of Cambridge, 1998.

[61] Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

[62] Tom Drummond and Roberto Cipolla. Application of lie algebra to visual servoing. *International Journal of Computer Vision*, 37(1):21–41, 2000.

[63] Pantelis Elinas and James J. Little. σmcl: Monte-carlo localization for mobile robots with stereo vision. In *Robotics: Science and Systems I*, Cambridge, Massachusetts, 2005.

[64] Bernard Espiau, François Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, June 1992.

[65] Yiu fai Wong. Cmac learning is governed by a single parameter. In *Proceedings of the 1993 IEEE International Conference on Neural Networks*, volume 3, pages 1439–1443, San Francisco, March 1993.

[66] Yiu fai Wong and Athanasios Sideris. Learning convergence in the cerebellar model articulation controller. *IEEE Transactions on Neural Networks*, 3(1):115–121, January 1992.

[67] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[68] John T. Feddema and Owen R. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Transactions on Robotics and Automation*, 5(5):691–700, October 1989.

[69] H.G. Feichtinger, C. Cenker, M Mayer, H. Steier, and T. Strohmer. New variants of the pocs method using affine subspaces of finite codimension with applications to irregular sampling. In *Proceedings of the SPIE Conference on Visual Communications and Image Processing*, volume 1818, pages 299–310, November 1992.

[70] Cornelia Fermüller. Global 3-d motion estimation. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 415 – 421, June 1993.

[71] Romeo Tatsambon Fomena and François Chaumette. Visual servoing from spheres using a spherical projection model. In *Proceedings of the 2007 IEEE Internatinal Conference on Robotics and Automation*, pages 2080–2085, Rome, April 2007.

[72] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*. AAAI, 1999.

[73] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

[74] Olac Fuentes and Randal C. Nelson. Learning dextrous manipulation skills for multifingered robot hands using the evolution strategy. *Machine Learning*, 1998.

[75] Nicholas R. Gans, Peter I. Corke, and Seth A. Hutchinson. Performance tests of partitioned approaches to visual servo control. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, volume 2, pages 1616–1623, May 2002.

[76] Chris Gaskett, Luke Fletcher, and Alexander Zelinsky. Reiforcement learning for a vision based mobile robot. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 403–409, November 2000.

[77] Chris Gaskett, Luke Fletcher, and Alexander Zelinsky. Reinforcement learning for visual servoing of a mobile robot. *Proceedings of the Australian Conference on Robotics and Automation*, 2000.

[78] Chris Gaskett, David Wettergreen, and Alexander Zelinsky. Reinforcement learning applied to the control of an autonomous underwater vehicle. *Proceedings of the Australian Conference on Robotics and Automation*, 1999.

[79] Christopher Geyer and Kostas Daniilidis. A unifying theory for central panoramic systems and practical applications. In *Proceedings of the 6th European Conference on Computer Vision-Part II*, volume 1843 of *Lecture Notes In Computer Science*, pages 445–461, 2000.

[80] Christopher Geyer and Kostas Daniilidis. Mirrors in motion: Epipolar geometry and motion estimation. In *Proceedings of the Ninth IEEE International Conference on Compuber Vision*, volume 2, pages 766–773, Nice, October 2003.

[81] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[82] M.A. Goodale, D. Pelisson, and C. Prablanc. Large adjustments in visually guided reaching do not depend on vision of the hand or perception of target displacement. *Nature*, 320:748–750, April 1986.

[83] Clément M. Gosselin and cois Hamel Jean-Fran The agile eye: a high performance three-degree-of-freedom camera-orienting device. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 781–786, San Diego, CA, May 2004.

[84] Gregory D. Hager. Calibration-free visual control using projective invariance. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 1009–1015, June 1995.

[85] Gregory D. Hager. A modular system for robust positioning using feedback from stero vision. *IEEE Transactions on Robotics and Automation*, 13(4):582–595, August 1997.

[86] Gregory D. Hager, Wen-Chung Chang, and A.S. Morse. Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 4, pages 2850–2856, May 1994.

[87] C. Harris. Geometry from visual motion. In A. Blake and A. Yuille, editors, *Active Vision*, pages 264–284. MIT Press, 1992.

[88] R.I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 19:580–593, June 1997.

[89] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[90] Koichi Hashimoto and Toshiro Noritsugu. Potential switching control in visual servo. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, pages 2765–2770, April 2000.

[91] Koichi Hashimoto and Toshiro Noritsugy. Performance and sensitivity in visual servoing. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, volume 3, pages 2321–2326, May 1998.

[92] Chao He, Lixin Xu, and Yuhe Zhang. Learning convergence of cmac algorithm. *Neural Processing Letters*, 14(1):61–74, August 2001.

[93] Donald O. Hebb. *The Organization of Behaviour*. John Wiley & Sons Inc, 1949.

[94] David J. Heeger and Allan D. Jepson. Subspace methods for recovering rigid motion i: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.

[95] Gabor T. Herman and Lorraine B. Meyer. Algebraic reconstruction techniques can be made computationally efficient. *IEEE Transactions on Medical Imaging*, 12(3):600–609, September 1993.

[96] J.P. Hespanha., Z. Dodds, G.D. Hager, and A.S. Morse. What tasks can be performed with an uncalibrated stereo vision system? *International Journal of Computer Vision*, 35(1):65–85, 1999.

[97] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. *Image and Vision Computing*, 12(3):187–192, April 1994.

[98] K. Hosada, K. Sakamoto, and M. Asada. Trajectory generation for obstacle avoidance of uncalibrated stereo visual servoing without 3d reconstruction. In *International Conference on Intelligent Robotics Systems*, pages 29–34, 1995.

[99] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM, 1991.

[100] Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[101] Braintech Inc. evisionfactory: the 3d vision guidance software platform. Brochure, 2005. http://www.braintech.com/.

[102] Michal Irani, Benny Rousso, and Shmuel Peleg. Recovery of ego-motion using region alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3), March 1997.

[103] M. Jagersand. Visual servoing using trust region methods and estimation of the full coupled visual-motor jacobian. In *IASTED Applications of Robotics and Control '96*, 1996.

[104] P. Jiang, P.-Y. Woo, and R. Unbehauen. Iterative learning control for manipulator trajectory tracking without any control singularity. *Robotica*, 2002.

[105] S. Kaczmarz. Angenäherte auflösung von systemen linearer gleichungen. *Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres*, pages 355–357, 1927.

[106] Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

[107] Rafael Kelly. Robust asymptotically stable visual servoing of planar robot. *IEEE Transactions on Robotics and Automation*, 5(12):759–766, October 1996.

[108] Rafael Kelly, Paul Shirkey, and Mark W. Spong. Fixed-camera visual servo control for planar robots. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, volume 3, pages 2643–2669, April 1996.

[109] W. Khalil and J.F. Kleinfinger. A new geometric notation for open and closed-loop robots. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pages 1174–1179, 1986.

[110] Sven Koenig and Reid G. Simmons. Passive distance learning for robot navigation. In *Proceedings ofthe Thirteenth International Conference on Machine Learning (ICML)*, pages 266–274, 1996.

[111] A.J. Koivo and Nasser Houshangi. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transactions on Systems, Man and Cybernetics*, 1991.

[112] R. Matthew Kretchmar and Charles W. Anderson. Comparison of cmacs and radial basis functions for local function approximators in reinforcement learning. In *Proceedings of the 1997 IEEE International Conference on Neural Networks*, volume 2, pages 834–837, June 1997.

[113] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):89–96, 1992.

[114] Simon Leonard, Ambrose Chan, Elizabeth A. Croft, and James J. Little. Robust motion generation for vision-guided robot bin-picking. In *Proceedings of IMECE2007. 2007 ASME International Mechanical Engineering Congress and Exposition*, Seattle, November 2007.

[115] Simon Leonard, Elizabeth A. Croft, and James J. Little. Dynamic visibility checking for vision-based motion planning. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 2283–2288, Los Angeles, California, May 2008.

[116] Simon Léonard and Martin Jägersand. Approximating the visuomotor function for visual servoing. In *Proceedings of 1st Canadian Conference on Computer and Robot Vision (CRV'04)*, pages 112–119, London, Canada, May 2004.

[117] Simon Léonard and Martin Jägersand. Learning based visual servoing. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'04)*, volume 1, pages 680–685, Sendai, Japan, September 2004.

[118] Simon Léonard and Martin Jägersand. Incremental learning for mapping image variations to actions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA'05)*, pages 4224–4229, Barcelona, Spain, April 2005.

[119] Simon Léonard and Martin Jägersand. Adaptive control for estimating translations from image-based variations. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA'06)*, pages 4106–4111, Orlando, U.S., May 2006.

[120] Simon Léonard and Martin Jägersand. On with the visuomotor function: A 6dof adaptive approach for modeling image-based variations and visual servoing. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA'07)*, pages 4106–4111, Rome, Italy, April 2007.

[121] Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, September 1998.

[122] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[123] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[124] John Y. S. Luh, Michael W. Walker, and Richard P. C. Paul. Resolved acceleration control of mechanical manipulators. *IEEE Transactions on Automatic Control*, 25(3):468–474, 1980.

[125] Sonja Macfarlane and Elizabeth A. Croft. Jerk-bounded manipulator trajectory planning: Design for real-time applications. *IEEE Transactions on Robotics and Automation*, 19(1):42–52, February 2003.

[126] Robert Mahony, Tarek Hamel, and François Chaumette. A decoupled image space approach to visual servo control of a robotic manipulator. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 3781–3786, May 2002.

[127] Ezio Malis. Visual servoing invariant to changes in camera intrinsic parameters. In *Proceedings of the Eighth International Conference on Computer Vision*, volume 1, pages 704–709, July 2001.

[128] Ezio Malis. Vision-based control invariant to camera intrinsic parameters: stability analysis and path tracking. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, volume 1, pages 217–222, May 2002.

[129] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the 2004 IEEE International Converence on Robotics and Automation*, pages 1843–1848, New Orleans, April 2004.

[130] Ezio Malis and Selim Benhimane. A unified approach to visual tracking and servoing. *Robotics and Autonomous Systems*, 52(1):39–52, July 2005.

[131] Ezio Malis and François Chaumette. 2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *International Journal of Computer Vision*, 37(1), June 2000.

[132] Ezio Malis, François Chaumette, and Sylvie Boudet. 2d 1/2 visual servoing stability analysis with respect to camera calibration errors. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots Systems*, volume 2, pages 691–697, October 1998.

[133] Ezio Malis, François Chaumette, and Sylvie Boudet. Positioning a coarse-alibrated camera with respect to an unknown object by 2d 1/2 visual servoing. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, pages 1352–1359, May 1998.

[134] Ezio Malis, François Chaumette, and Sylvie Boudet. 2-1/2-d visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, April 1999.

[135] Éric Marchand and François Chaumette. Virtual visual servoing: a framework for real-time augmented reality. In G. Drettakis and H.-P. Seidel, editors, *EUROGRAPHICS*, volume 21, 2002.

[136] M. Marjanović, B. Scassellati, and M. Wiiliamson. Self-taught visually-guided pointing for a humanoid robot. In *Proceedings of the Conference on Simulation of Adaptive Behavior*, pages 35–44, 1996.

[137] David Marr. *Vision*. Freeman, 1982.

[138] Philippe Martinet and Jean Gallice. Position based visual servoing using a non-linear approach. In *Proceedings of the 1999 IEEE/RSJ Internatinal Conference on Intelligent Robots and Systems*, pages 531–536, Kyongju, Korea, October 1999.

[139] Geoffrey McLachlan and David Peel. *Finite Mixture Models*. Wiley series in probability and statistics. John Wiley & Sons Inc., 2000.

[140] B. Mehta and S. Schall. Forward models in visuomotor control. *Journal of Neurophysiology*, 88:942–953, 2002.

[141] Youcef Mezouar and François Chaumette. Path planning for robust image-based control. *IEEE Transactions on Robotics and Automation*, 18(4):534–549, August 2002.

[142] Youcef Mezouar, Anthony Remazelles, Patrick Gros, and François Chaumette. Images interpolation for image-based control under large displacement. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, volume 4, pages 3787–3794, May 2002.

[143] A.D. Milner and M.A. Goodale. *Visual Brain In Action*. Number 27 in Oxford Psychology Series. Oxford University Press, 1995.

[144] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[145] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[146] Akio Namiki, Yoshihiro Nakabo, Idaku Ishii, and Masatoshi Ishikawa. 1-ms sensory-motor fusion system. *IEEE/ASME Transactions on Mechatronics*, 5(3):244–252, September 2000.

[147] Frank Natterer. *The Mathematics of Computerized Tomography*. SIAM: Society for Industrial and Applied Mathematics, 2001.

[148] Paul Y. Oh and Peter K. Allen. Visual servoing by partitioning degrees of freedom. *IEEE Transactions on Robotics and Automation*, 17(1):1–17, Febuary 2001.

[149] Serge J. Olszanskyj, James M. Lebak, and Adam W. Bojanczyk. Rank-k modification methods for recursive least squares problems. *Numerical Algorithms*, 7:325–354, 1994.

[150] M. Pagel, E. Maël, and C. Von Der Malsburg. Self calibration of the fixation movement of a stereo camera head. *Machine Learning*, 31:161–186, 1998.

[151] Nikolaos P. Papanikolopoulos and Pradeep K. Khosla. Adaptive robot visual tracking: Theory and experiments. *IEEE Transactions on Automatic Control*, 38(3):429–445, 1993.

[152] Nikolaos P. Papanikolopoulos, Pradeep K. Khosla, and Takeo Kanade. Visual tracking of a moving target by a camera mounted on a robot: A combination of control and vision. *IEEE Transactions on Robotics and Automation*, 9(1):14–35, February 1993.

[153] Jae Seok Park and Myung Jin Chung. Path planning with uncalibrated stereo rig for image-based visual servoing under large pose discrepancy. *IEEE Transactions on Robotics and Automation*, 19(2):250–258, April 2003.

[154] P.C. Parks and J. Militzer. Convergence properties of associative memory storage for learning control systems. *Automation and Remote Control*, 50(2):254–286, 1989.

[155] P.C. Parks and J. Militzer. A comparaison of five algorithms for the training of cmac memories for learning control systems. *Automatica*, 28(5):1027–1035, September 1992.

[156] J. Pauli. Learning to recognize and grasp objects. *Machine Learning*, 1998.

[157] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[158] Ting Qin, Zonghai Chen, Haitao Zhang, Sifu Li, Wei Xiang, and Ming Li. A learning algorithm of cmac based on rls. *Neural Processing Letters*, 19(1):49–61, December 2004.

[159] Ting Qin, Haitao Zhang, Zonghai Chen, and Wei Xiang. Continuous cmac-qrls and its systolic array. *Neural Processing Letters*, 22(1):1–16, August 2005.

[160] Fernando Reyes and Rafael Kelly. Experimental evaluation of fixed-camera direct visual controllers on a direct-drive robot. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, volume 3, pages 2327–2332, May 1998.

[161] Frank Rosenblatt. *Principles of neurodynamics;: Perceptrons and the theory of brain mechanisms*. Spartan Books, 1962.

[162] A. Ruf and R. Horaud. Visual servoing of robot manipulator part i: Projective kinematics. *International Journal of Robotics Research*, 1999.

[163] M. Salganicoff, L.H. Ungar, and Bajcsy R. Active learning for vision-based robot grasping. *Machine Learning*, 1996.

[164] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control The Task Function Approach*, volume 22 of *Oxford Engineering Science Series*. Oxford University Press, 1991.

[165] Florian Schramm, Franck Geffard, Guillaume Morel, and Alain Micaelli. Calibration free image point path planning simultaneously ensuring visibility and controlling camera path. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2074–2079, Roma, Italy, April 2007.

[166] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, 21(8):735–758, August 2002.

[167] Masoud Shahamiri and Martin Jagersand. Uncalibrated visual servoing using a biased newton method for on-line singularity detection and avoidance. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005*, pages 3953–3958, 2005.

[168] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. The MIT Press, 2004.

[169] Reid Simmons and Sven Koenig. Probabilistic navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[170] Christopher E. Smith and Nikolaos P. Papanikolopoulos. Computation of shape through controlled active exploration. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, volume 3, pages 2516–2521, May 1994.

[171] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationship in robotics. In I.J. Cox and G.T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[172] Jay Stavnitzky and David Capson. Multiple camera model-based 3-d visual servo. *IEEE Transactions on Robotics and Automation*, 16(6):732–739, December 2000.

[173] Thomas Strohmer and Roman Vershynin. A randomized solver for linear systems with exponential convergence. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4110 of *Lecture Notes in Computer Science*, pages 499–507. Springer Berlin, 2006.

[174] Rchard S. Sutton. Generalization in reinforcement learning: Successful examples using sparce coarse coding. In *Advances in Neural Information Processing Systems 8*, pages 1038–1044. The MIT Press, 1995.

[175] Richard S. Sutton and Steven D. Whitehead. Online learning with random representations. In *Proceedings of the 10th International Conference on Machine Learning*, pages 314–321, Amherst, June 1993. Morgan Kaufmann.

[176] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. The MIT Press, 1999.

[177] Omar Tahri, François Chaumette, and Youcef Mezouar. New decoupled visual servoing scheme based on invariants from projection onto a sphere. In *Proceedings of the 2008 IEEE Internatinal Conference on Robotics and Automation*, Pasadena, May 2008.

[178] Yasutake Takahashi, Masanori Takeda, and Minoru Asada. Continuous valued q-learning for vision-guided behavior acquisition. *Proceedings of the 1999 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 255–260, August 1999.

[179] Sebastian Thrun. Bayesian landmark learning for mobile robot localization. *Machine Learning*, 33:41–76, 1998.

[180] Benoit Thuilot, Philippe Martinet, Lionel Cordesses, and Jean Gallice. Position based visual servoing : keeping the object in the field of vision. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1624–1629, Washington, DC, May 2002.

[181] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[182] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[183] Markus Vincze. Dynamics and system performance of visual servoing. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, volume 1, pages 644–649, April 2000.

[184] Lee E. Weiss, Arthur C. Sanderson, and Charles P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3(5):404–417, October 1987.

[185] D. Adrian Wilkinson. Visual-motor control loop: A linear system? *Journal of Experimental Psychology*, 89(2):250–257, August 1971.

[186] William J. Wilson, Carol C. Williams Hulls, and Graham S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions or Robotics and Automation*, 12(5):684–696, October 1996.

[187] Patrick Winston. Progress in vision and robotics. Technical Report AI TR-281, M.I.T., May 1973.

[188] Hanqi Zhuang, Zvi S. Roth, and R. Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form $ax = yb$. *IEEE Transactions on Robotics and Automation*, 10(4):549–554, August 1994.

Appendix A

Appendix

A.1 Visuomotor Function 6DOF

To solve for the 6DOF motion, Equation 3.42 is expressed in terms of ZYX Euler angles

$$\mathbf{f}_{6DOF}(\boldsymbol{\theta}, \mathbf{x}) = \begin{bmatrix} f_x(\boldsymbol{\theta}, \mathbf{x}) \\ f_y(\boldsymbol{\theta}, \mathbf{x}) \end{bmatrix} = V \begin{bmatrix} \cos(\alpha)\cos(\beta) \\ \sin(\alpha)\cos(\beta) \\ -\sin(\beta) \\ -\sin(\alpha)\cos(\phi) + \cos(\alpha)\sin(\beta)\sin(\phi) \\ \cos(\alpha)\cos(\phi) + \sin(\alpha)\sin(\beta)\sin(\phi) \\ \cos(\beta)\sin(\phi) \\ \sin(\alpha)\sin(\phi) + \cos(\alpha)\sin(\beta)\cos(\phi) \\ -\cos(\alpha)\sin(\phi) + \sin(\alpha)\sin(\beta)\cos(\phi) \\ \cos(\beta)\cos(\phi) \\ \mathbf{t} \\ 1 \end{bmatrix} - \mathbf{e} = 0. \quad \text{(A.1)}$$

Given that the elements in V are labeled $v_{i,j}$, the elements of the Jacobian J_f are defined as

follow

$$
\begin{aligned}
\frac{\partial J_{f_z}}{\partial \alpha} &= -v_{1,1}\sin(\alpha)\cos(\beta) + v_{1,2}\cos(\alpha)\cos(\beta) \\
&\quad + v_{1,4}(-\cos(\alpha)\cos(\phi) - \sin(\alpha)\sin(\beta)\sin(\phi)) \\
&\quad + v_{1,5}(-\sin(\alpha)\cos(\phi) + \cos(\alpha)\sin(\beta)\sin(\phi)) \\
&\quad + v_{1,7}(\cos(\alpha)\sin(\phi) - \sin(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{1,8}(\sin(\alpha)\sin(\phi) + \cos(\alpha)\sin(\beta)\cos(\phi)) \\
\frac{\partial J_{f_z}}{\partial \beta} &= -v_{1,1}\cos(\alpha)\sin(\beta) - v_{1,2}\sin(\alpha)\sin(\beta) \\
&\quad - v_{1,3}\cos(\beta) + v_{1,4}\cos(\alpha)\cos(\beta)\sin(\phi) \\
&\quad + v_{1,5}\sin(\alpha)\cos(\beta)\sin(\phi) - v_{1,6}\sin(\beta)\sin(\phi) \\
&\quad + v_{1,7}\cos(\alpha)\cos(\beta)\cos(\phi) + v_{1,8}\sin(\alpha)\cos(\beta)\cos(\phi) \\
&\quad - v_{1,9}\sin(\beta)\cos(\phi) \\
\frac{\partial J_{f_z}}{\partial \phi} &= v_{1,4}(\sin(\alpha)\sin(\phi) + \cos(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{1,5}(-\cos(\alpha)\sin(\phi) + \sin(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{1,6}\cos(\beta)\cos(\phi) + v_{1,7}(\sin(\alpha)\cos(\phi) \\
&\quad - \cos(\alpha)\sin(\beta)\sin(\phi)) + v_{1,8}(-\cos(\alpha)\cos(\phi) \\
&\quad - \sin(\alpha)\sin(\beta)\sin(\phi)) - v_{1,9}\cos(\beta)\sin(\phi) \\
\frac{\partial J_{f_z}}{\partial t_x} &= v_{1,10} \\
\frac{\partial J_{f_z}}{\partial t_y} &= v_{1,11} \\
\frac{\partial J_{f_z}}{\partial t_z} &= v_{1,12}
\end{aligned}
$$

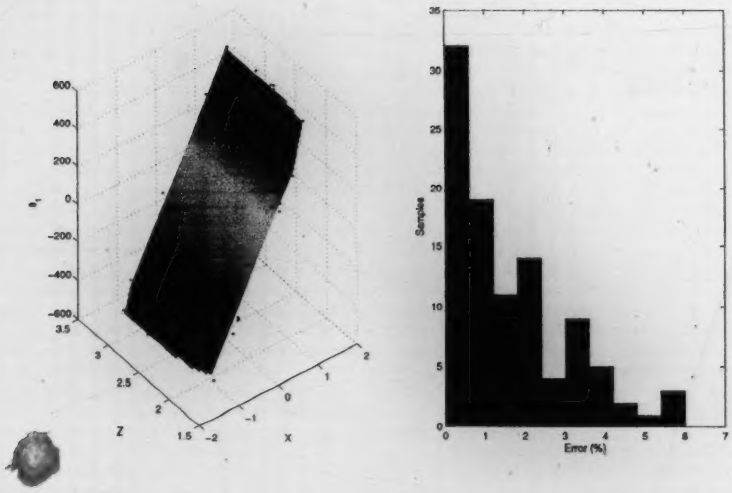


Figure A.1: Comparaison between approximations of θ_1 and analytical values. Distribution of relative absolute errors.

$$
\begin{aligned}
\frac{\partial J_{f_y}}{\partial \alpha} &= -v_{2,1}\sin(\alpha)\cos(\beta) + v_{2,2}\cos(\alpha)\cos(\beta) \\
&\quad + v_{2,4}(-\cos(\alpha)\cos(\phi) - \sin(\alpha)\sin(\beta)\sin(\phi)) \\
&\quad + v_{2,5}(-\sin(\alpha)\cos(\phi) + \cos(\alpha)\sin(\beta)\sin(\phi)) \\
&\quad + v_{2,7}(\cos(\alpha)\sin(\phi) - \sin(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{2,8}(\sin(\alpha)\sin(\phi) + \cos(\alpha)\sin(\beta)\cos(\phi)) \\
\frac{\partial J_{f_y}}{\partial \beta} &= -v_{2,1}\cos(\alpha)\sin(\beta) - v_{2,2}\sin(\alpha)\sin(\beta) - v_{2,3}\cos(\beta) \\
&\quad + v_{2,4}\cos(\alpha)\cos(\beta)\sin(\phi) + v_{2,5}\sin(\alpha)\cos(\beta)\sin(\phi) \\
&\quad - v_{2,6}\sin(\beta)\sin(\phi) + v_{2,7}\cos(\alpha)\cos(\beta)\cos(\phi) \\
&\quad + v_{2,8}\sin(\alpha)\cos(\beta)\cos(\phi) - v_{2,9}\sin(\beta)\cos(\phi) \\
\frac{\partial J_{f_y}}{\partial \phi} &= v_{2,4}(\sin(\alpha)\sin(\phi) + \cos(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{2,5}(-\cos(\alpha)\sin(\phi) + \sin(\alpha)\sin(\beta)\cos(\phi)) \\
&\quad + v_{2,6}\cos(\beta)\cos(\phi) + v_{2,7}(\sin(\alpha)\cos(\phi) - \cos(\alpha)\sin(\beta)\sin(\phi)) \\
&\quad + v_{2,8}(-\cos(\alpha)\cos(\phi) - \sin(\alpha)\sin(\beta)\sin(\phi)) - v_{2,9}\cos(\beta)\sin(\phi) \\
\frac{\partial J_{f_y}}{\partial t_x} &= v_{2,10} \\
\frac{\partial J_{f_y}}{\partial t_y} &= v_{2,11} \\
\frac{\partial J_{f_y}}{\partial t_z} &= v_{2,12}
\end{aligned}
$$

A.2 CMAC: Approximation of the Visuomotor Parameters

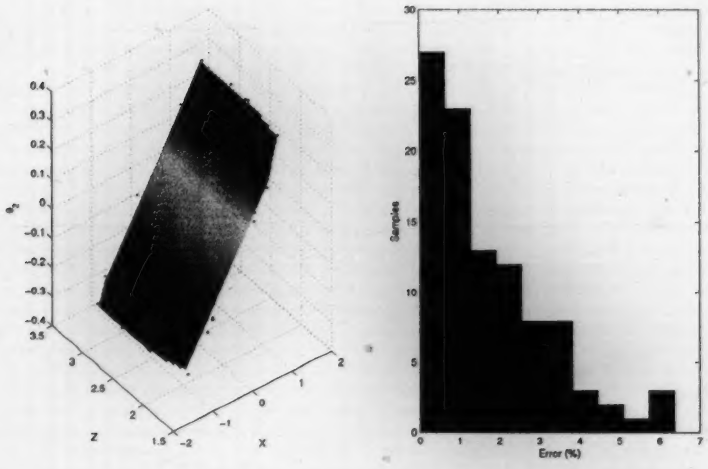


Figure A.2: Comparaison between approximations of θ_2 and analytical values. Distribution of relative absolute errors.

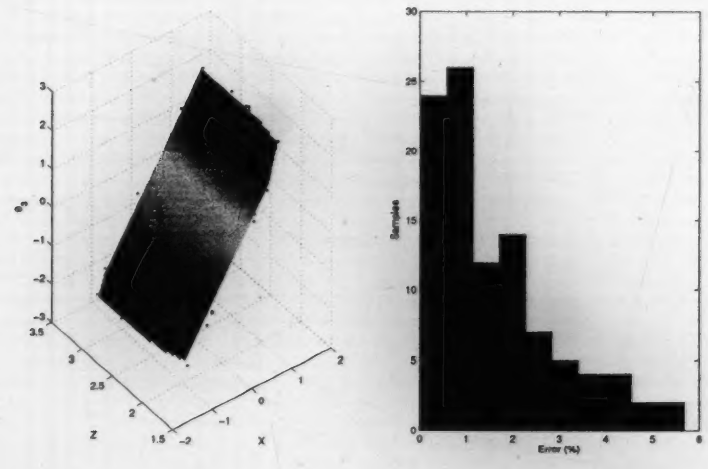


Figure A.3: Comparaison between approximations of θ_3 and analytical values. Distribution of relative absolute errors.

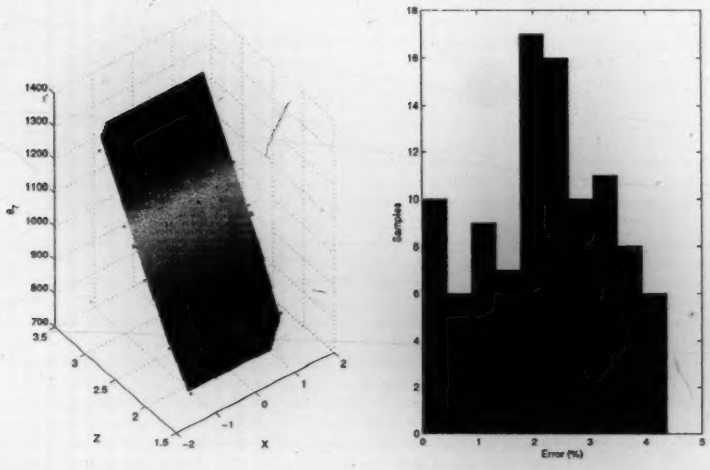


Figure A.4: Comparaison between approximations of θ_7 and analytical values. Distribution of relative absolute errors.

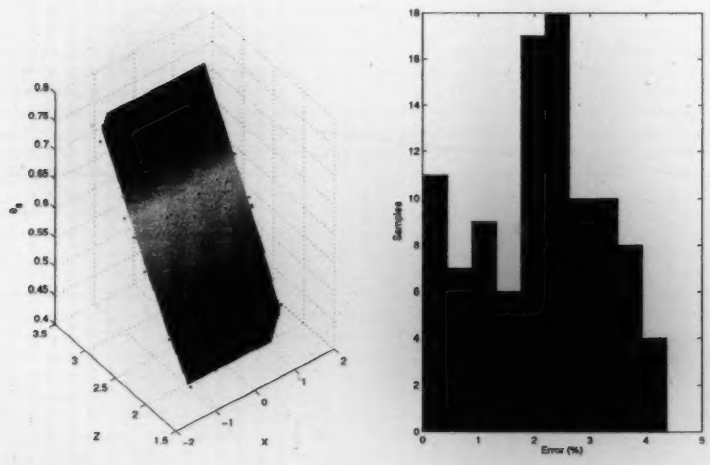


Figure A.5: Comparaison between approximations of θ_8 and analytical values. Distribution of relative absolute errors.

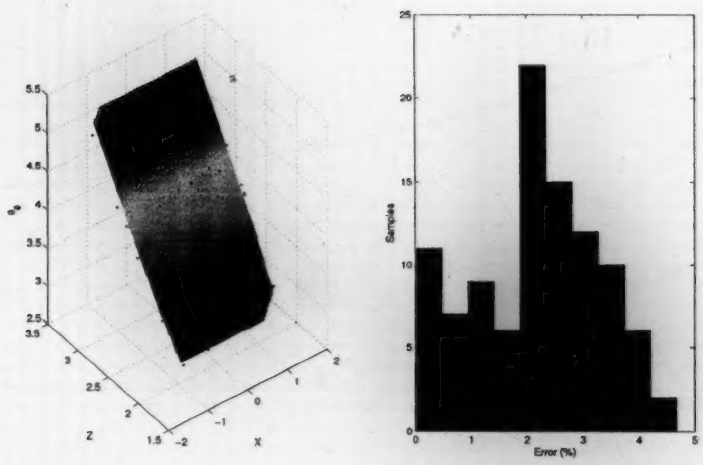


Figure A.6: Comparaison between approximations of θ_9 and analytical values. Distribution of relative absolute errors.

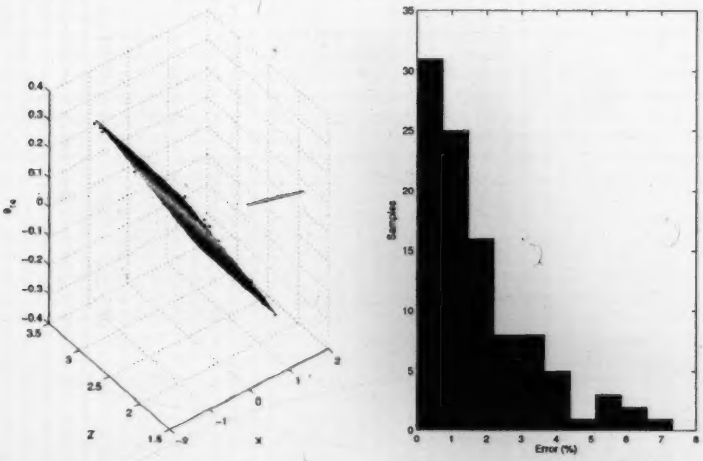


Figure A.7: Comparaison between approximations of θ_{14} and analytical values. Distribution of relative absolute errors.

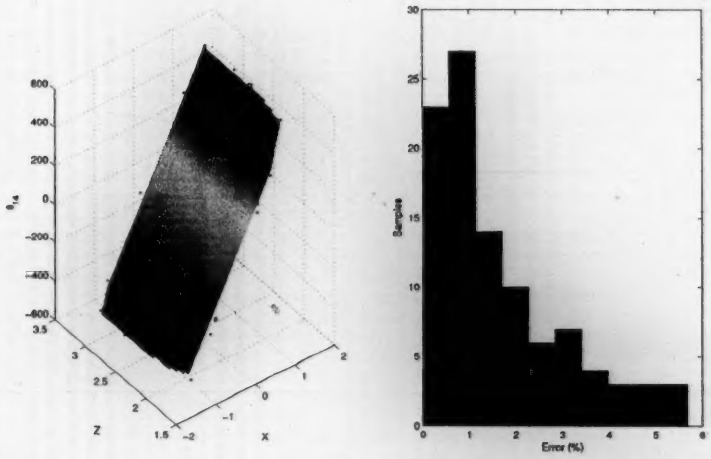


Figure A.8: Comparaison between approximations of θ_{15} and analytical values. Distribution of relative absolute errors.

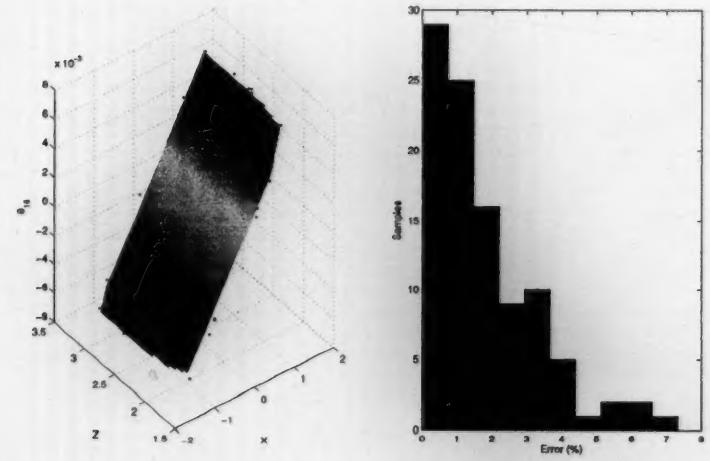


Figure A.9: Comparaison between approximations of θ_{16} and analytical values. Distribution of relative absolute errors.

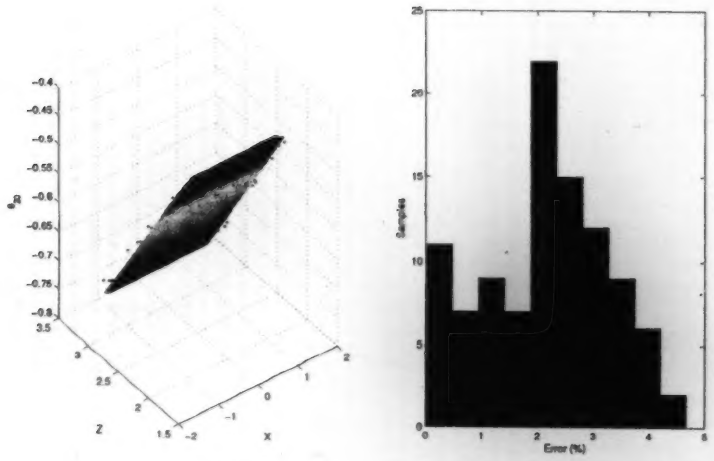


Figure A.10: Comparaison between approximations of θ_{20} and analytical values. Distribution of relative absolute errors.

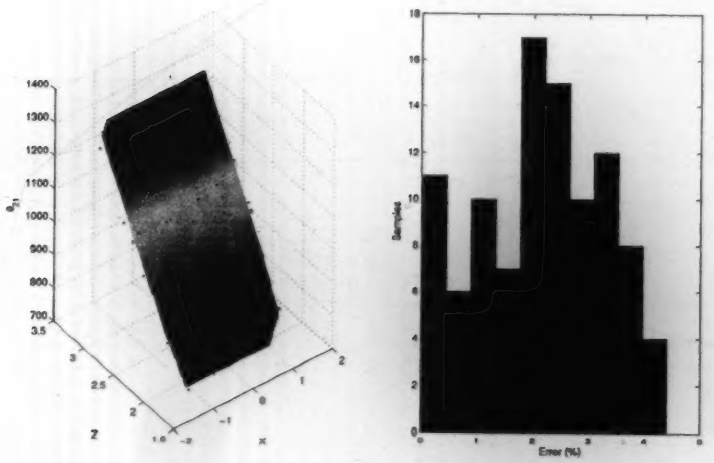


Figure A.11: Comparaison between approximations of θ_{21} and analytical values. Distribution of relative absolute errors.

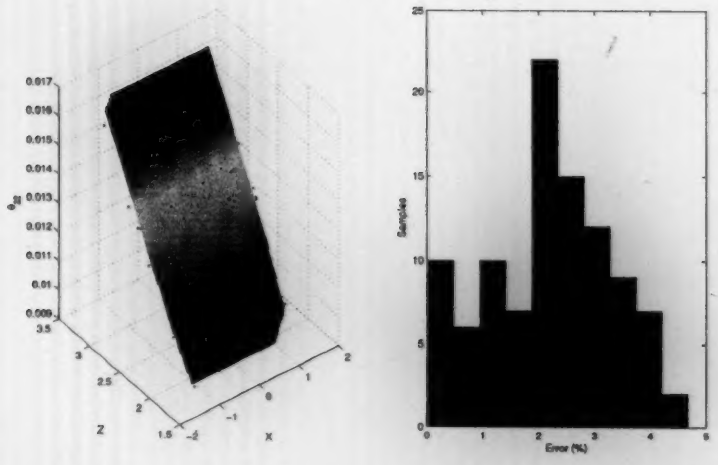


Figure A.12: Comparaison between approximations of θ_{22} and analytical values. Distribution of relative absolute errors.

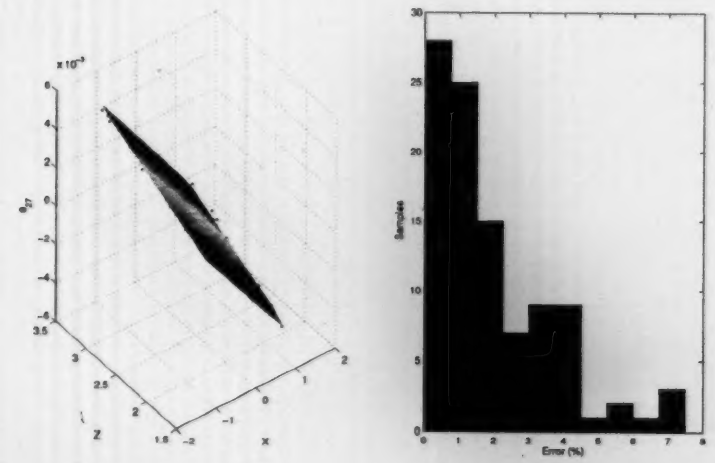


Figure A.13: Comparaison between approximations of θ_{27} and analytical values. Distribution of relative absolute errors.

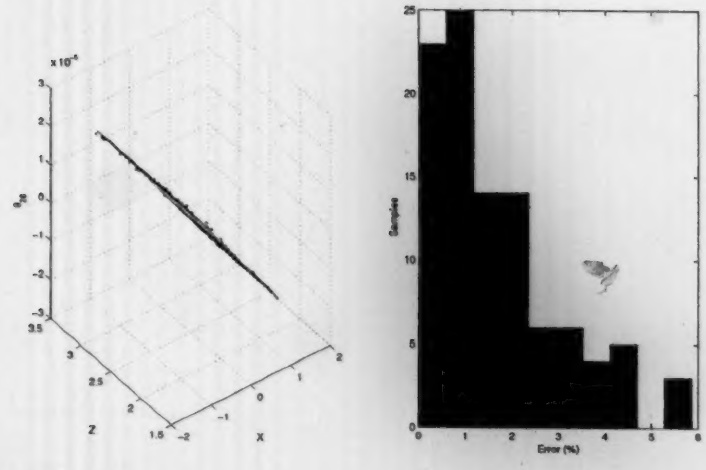


Figure A.14: Comparaison between approximations of θ_{28} and analytical values. Distribution of relative absolute errors.

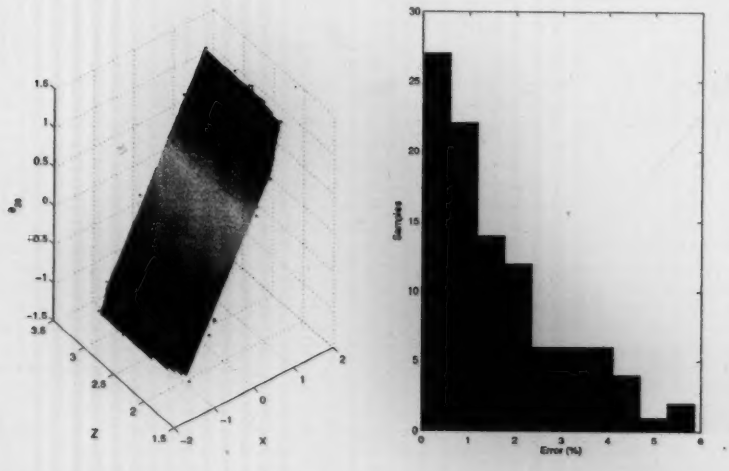


Figure A.15: Comparaison between approximations of θ_{29} and analytical values. Distribution of relative absolute errors.

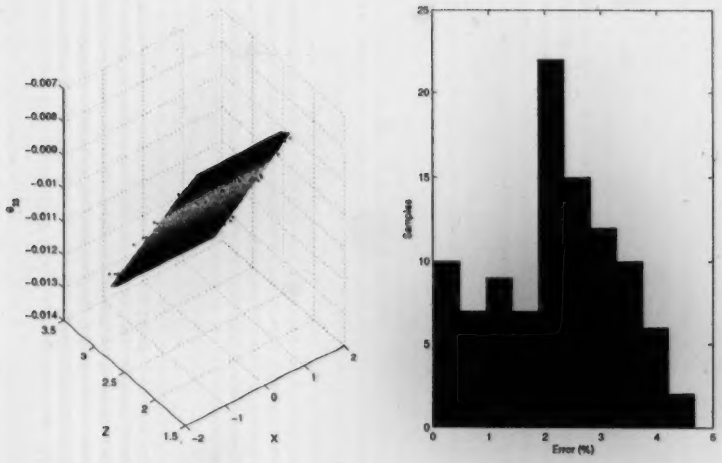


Figure A.16: Comparaison between approximations of θ_{33} and analytical values. Distribution of relative absolute errors.

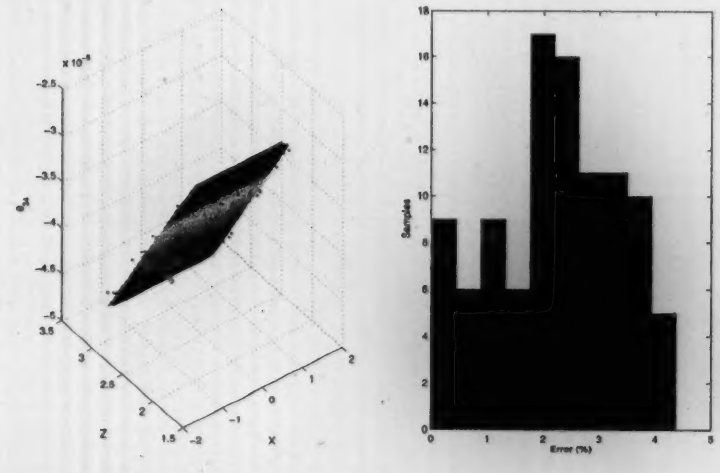


Figure A.17: Comparaison between approximations of θ_{34} and analytical values. Distribution of relative absolute errors.

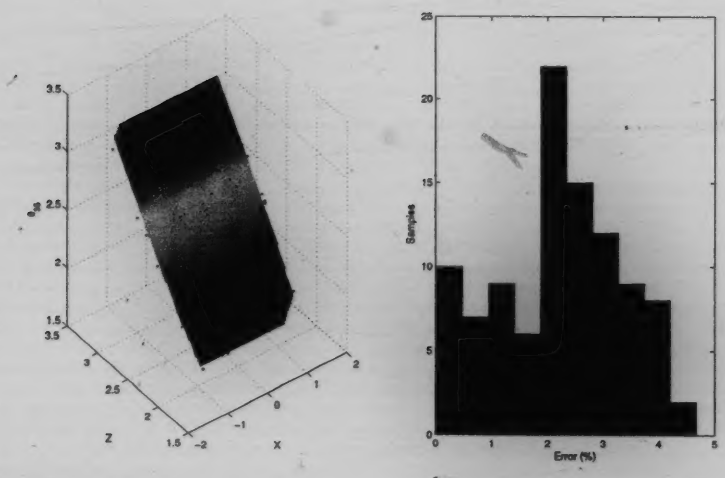


Figure A.18: Comparaison between approximations of θ_{35} and analytical values. Distribution of relative absolute errors.